VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zátěžové testování a optimalizace webového rozhraní systému Floreon+

Performance Testing and Optimization of the Floreon+ Web-based Interface

2015 Jan Křenek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jan Křenek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Zátěžové testování a optimalizace webového rozhraní systému Floreon+

Performance Testing and Optimization of the Floreon+ web-based Interface

Zásady pro vypracování:

Systém Floreon+ je modulárním systémem pro předpověď povodní a podporu krizového řízení. V současné době probíhá vývoj nového webového rozhraní tohoto systému s mapovovou komponentou a díky novým projektům a jejich medializaci se očekává nárůst uživatelů z řad bezpečnostních složek i veřejnosti, kteří budou toto webové rozhraní používat.
Cílem této práce je otestovat chování systému pro vysoký počet uživatelů a implementovat opatření, která zajistí jeho plynulý běh i při vysoké zátěži.

Konkrétní body zadání jsou:
1. Seznámit se se současným stavem připravovaného webového rozhraní systému Floreon+ a jeho technologiemi.
2. Vytvořit testovací plán a provést zátěžové testy tohoto webového systému pro vysoký počet uživatelů.
3. Navrhnout a implementovat vybrané optimalizační úpravy, které sníží dobu zpracování požadavků při vysokém počtu uživatelů.
4. Zhodnotit výslednou optimalizaci.

Seznam doporučené odborné literatury:

[1] Molyneaux, Ian. 2009. The Art of Application Performance Testing: Help for Programmers and Quality Assurance. O'Reilly Media, Inc.
[2] JQuery:
http://jquery-navod.cz/serial/serial-o-jquery
http://www.w3schools.com/jquery/
[3] OpenLayers:
http://workshops.boundlessgeo.com/openlayers-intro/
http://trac.osgeo.org/openlayers/wiki/Documentation
[4] WMS a WFS:
http://www.opengeospatial.org/standards/wms
http://www.opengeospatial.org/standards/wfs

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Štěpán Kuchař**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 5. května 2015 .....

Nejprve bych rád poděkoval vedoucímu své diplomové práce Ing. Štěpánu Kuchařovi, který mi během vypracování přispíval cennými radami, díky nimž jsem mohl vypracovat svou práci. Dále bych chtěl poděkovat i ostatním spolupracovníkům z projektu Floreon+, za pomoc a podnětné připomínky. Děkuji také vedení a pracovníkům centra IT4Inovations za zajištění příjemného pracovního prostředí a vstřícného přístupu v průběhu zpracovavání mé bakalářské práce.

Abstrakt

Cílem práce je otestovat chování webového rozhraní systému Floreon+ pro vysoký počet uživatelů a implementovat opatření, která zajistí jeho plynulý běh i při vysoké zátěži. Součástí práce bylo vytvoření testovacího balíku, který obsahuje testovací plány použité při testování webového rozhraní systému Floreon+.Testovací balík byl navržen tak, aby mohl být použit i pro testování webových rozhraní jiných projektů, které jsou založeny na podobném principu jako webové rozhraní systému Floreon+. Pro provádění testů byly zvoleny testovací nástroje JMeter a Visual Studio, které otestovaly webové rozhraní a nalezly úzká místa systému Floreon+. Při optimalizacích založených na provedeném testování byla ze systému odstraněna chybná volání, byly vytvořeny cachovací mechanismy pro zkrácení doby odpovědi a byla navržena distribuovaná architektura nasazení systému.

Klíčová slova: Zátěžové testování, Optimalizace, JMeter, Visual Studio, systém Floreon+

Abstract

The aim of the thesis is to test the behaviour of the Floreon+ system web interface for a large number of users and implement measures to ensure its smooth operation even at a high load. Part of the paper was the development of a test suite which contains testing plans used in testing the web interface of the Floreon+ system. The test suite was designed so that it could also be used for testing web interfaces of other projects which are based on a principle similar to that of the Floreon+ system web interface. Testing tools JMeter and Visual Studio were chosen to test the web interface and identify performance bottlenecks in the Floreon+ system. Within the course of optimizations based on the testing performed, error messages were eliminated from the system, caching mechanisms were developed to reduce the response time and a distributed deployment architecture was designed.

Keywords: Load testing, Optimization, JMeter, Visual Studio, Floreon+

Seznam použitých zkratek a symbolů

AJAX	–	Asynchronous JavaScript and XML
CLR	–	Common Language Runtime
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model
FTP	–	File Transfer Protocol
HTML	–	Hyper Text Markup Language
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
IIS	–	Internet Information Services
JPEG	–	Joint Photographic Experts Group
JS	–	JavaScript
JSON	–	JavaScript Object Notation
MS	–	Microsoft
PNG	–	Portable Network Graphics
POP3	–	Post Office Protocol
REST	–	Representational state transfer
RODOS	–	Rozvoj dopravních systémů
SMTP	–	Simple Mail Transfer Protocol
SOAP	–	Simple Object Access
SPA	–	Stupeň povodňové aktivity
SQL	–	Structured Query Language
TIFF	–	Tagged Image File Format
WCF	–	Windows Communication Foundation
WFS	–	Web Feature Service
WMS	–	Web Map Service
XML	–	Extensible Markup Language

Obsah

Seznam tabulek

Seznam obrázků

1 Úvod

Testování popisuje jednu z oblastí vývoje software, která by neměla být opomíjena v žádném projektu, protože zanedbání této části může firmě přinést špatnou reputaci na trhu nebo přinést větší finanční náklady při úpravách software. Z tohoto důvodu by se na tuto oblast nemělo zapomínat a mělo by se jí věnovat dostatek času. Kvalita používaného software je jednou z hlavních konkurenčních výhod. Proto bylo vytvořeno mnoho standardů zabývajících se dosažením kvality a nástrojů pro efektivnější testování softwaru. Testování by se mělo provádět po každé etapě vývoje systému v softwarových procesech (Vodopádový model, Scrum, Extrémní programování a dalších) pro odhalení defektů, chyb či selhání systému.

Seznámení s webovým rozhraním systému Floreon+ a použitými technologiemi na systému Floreon+ naleznete v kapitole 2 Seznámení se systémem Floreon+ a použitými technologiemi.

Daný problém se zabývá jen jednou částí testování - zátěžovým testováním implementovaného softwaru. Jako vhodné programy pro zátěžové testování se osvědčily JMeter a Visual Studio a pro následnou reprezentaci výsledku Microsoft Excel. O problematice se více dozvíte v kapitole 3 Zátěžové testování a testovací plán.

Cílem dokumentu není jen vykonání zátěžového testování webového rozhraní systému Floreon+, ale také navržení úprav, které sníží přístupové doby webového rozhraní a zvýší možnost přístupu více uživatelům na webové rozhraní. Na základě výsledků testů byla identifikována úzká místa systému a ta byla optimalizována za účelem zrychlení systému. Více o úpravách se dočtete v kapitole 4 Úpravy a optimalizace webového rozhraní.

Výsledné zhodnocení zátěžových testů a optimalizací, úspěšných i neúspěšných, nad webovým rozhraním Floreon+ naleznete v kapitole 5 Zhodnocení výsledků testů a optimalizací.

Kapitola 6 Závěr obsahuje celkové shrnutí testování. Zároveň navrhuje rozšíření o testovací plány, které v průběhu testování nemohly být provedeny, protože byly v době tvorby testovacích plánů a provádění testování pouze rozpracovány.

2 Seznámení se systémem Floreon+ a použitými technologiemi

Kapitola pojednává o problémech, které systém Floreon+ pomáhá řešit.

2.1 Co je to systém Floreon+?

Floreon+ je integrovaný systém, poskytující řešení monitorování, modelování, podporu a řešení krizových situací s aktuálním zaměřením na Moravskoslezský kraj a možným rozšířením pro celou Českou republiku. Jedním z hlavních důvodů vzniku systému je výpočet a zobrazení aktuálních stavů vodních toků i předpovědi možných scénářů nebezpečí při intenzivních srážkách. Systém byl kromě zobrazení hydrologických informací rozšířen na dopravní informace a řešení úniku nebezpečných látek. Díky snadné integraci dat a zvolené technologii je systém možno jednoduše rozšiřovat o nové funkce. Z důvodu snadného použití, jednoduché manipulace, nenáročné integrace a zobrazení dat uživatelům je rozhraní systému řešeno webovým rozhraním s mapovými vrstvami [1].

2.2 Systém aktuálně umožňuje:

2.2.1 Srážko-odtokové modelování

Srážko-odtokové modely popisují reakci povodí na aktuální srážky. Díky systému Floreon+ je možné zobrazit aktuální stavy a predikované stavy průtoků ve vybraných měřících stanicích v povodí. Na základě získaných informací systém varuje v případě překročení stupňů povodňové aktivity. Predikovaný stav lze díky vypočteným simulacím zobrazit ve formě hydrogramů, a tak poskytnout včasnou informaci pro podporu rozhodování v krizových procesech [1].

2.2.2 Výpočet rozlivu a záplavových zón

Pro umožnění predikce rozlivů jsou potřebné výsledky predikce srážko-odtokových modelů na modelovaném území. Je tedy vhodné časově zařadit přípravu hydrodynamických modelů až po dokončení výpočtů tvorby srážko-odtokových modelů. Díky těmto modelům je možné vypočítat a zobrazit informace o průtocích, výšce vodní hladiny a rychlosti na příčných a podélných profilech a vizualizovat tak simulovaný rozliv do krajiny včetně jeho časového vývoje [1].

2.2.3 Modelování úniku nebezpečných látek

Systém Floreon+ je schopný simulovat šíření nebezpečných látek v důsledku havárie a pomocí získaných dat vytvořit model proudění v ovzduší. Na základě tohoto modelu je možné předpovědět rozsah šíření a koncentraci nebezpečné látky v oblasti a zobrazit simulaci šíření ze zdroje. V systému Floreon+ je zohledněna i aktuální meteorologická

situace, která může významně ovlivnit vývoj havárie v čase. Tyto simulace umožňují efektivnější evakuaci oblasti a případnou likvidaci havárie.[1]

2.2.4 Sledování aktuální dopravní situace

Floreon+ je ve spolupráci s Centrem kompetence RODOS schopen monitorovat a zobrazovat aktuální dopravní situaci včetně informací o průjezdnosti cest a omezeních v dopravní síti. Velkou výhodou spojení všech dat do jednoho mapového prostředí je možnost snadné analýzy provázanosti jednotlivých modelovaných oblastí [1].

2.3 Technologie použité v systému Floreon+

Tato podkapitola řeší použité technologie na webovém rozhraní systému Floreon+. Jelikož se jedná o webové rozhraní, je pochopitelné, že komunikace bude prováděna mezi serverem a klientem pomocí HTTP protokolu. Technologie lze tudíž rozdělit na technologie běžící na straně klienta a technologie běžící na straně serveru.

2.3.1 Strana Serveru

Jelikož je ve webovém rozhraní požadováno zobrazení mapy, na serverové části systému je použit Geoserver. Geoserver dle publikované služby (WMS nebo WFS v našem případě) si z požadavku od klienta vezme parametry jako souřadnice, čas a velikost obrázku. Podle parametrů daných dotazem sestaví Geoserver tile (obrázek) z prostorových dat a pošle ho klientovi. Na webovém rozhraní systému Floreon+ je nezbytná existence přihlašování uživatelů a uchovávání jejich dat. Pro tuto funkcionalitu je možno používat relační databázi, kde jsou uložena data o uživatelích. WCF služba zajišťuje přihlašování a registraci uživatele. Pro výpočet různých simulací jsou použita data z relační databáze. Data jsou vykreslena a následně zaslána a zobrazena klientovi. Z důvodu oddělení lokální sítě od internetu a rozdělení zátěže mezi více serverů je použit proxy server.

Geoserver je opensourcový serverový nástroj napsaný v programovacím jazyce Java, který poskytuje velkou flexibilitu při tvorbě a sdílení dat. Geoserver je založen na poskytování geografických služeb s otevřenými standardy jako WFS, WMS a WCS. Geoserver slouží k publikaci geoprostorových dat do aplikací jako Google Earth nebo OpenLayers pro zobrazení dat ve webových prohlížečích [2], [3], [4].

PostgreSQL je výkonná opensourcová objektově relační databáze. Běží na všech hlavních operačních systémech, včetně systémů Linux, UNIX (SGI IRIX, Mac OS X, Solaris a další) a Windows. Kromě klasických datových typů podporuje ukládání velkých binárních objektů, včetně obrázků, videí nebo zvuků. Má plnou podporu cizích klíčů, spojení, pohledů, triggerů a uložených procedur [5].

PostGIS je rozšířením PostgreSQL. PostGIS přidává další geoprostorové datové typy a funkce pro jednodušší a výkonnější manipulaci s prostorovými daty. Většinou se používá pro uložení mapových dat. I OpenStreetMap používá PostGIS databázi pro ukládání map [6], [7].

MS SQL Server je relační databáze, která umožňuje ukládání základních datových typů. Umí vytvářet pohledy, triggery, procedury a spojení. V systému Floreon+ je PostgreSQL použita pro ukládání prostorových dat a MS SQL je aplikována pro ukládání dat uživatelského nastavení a dat potřebných pro modelování a simulace [8].

WCF je Framework zajišťující komunikaci mezi aplikacemi a umožňuje vytvářet aplikace orientované na služby. Zpráva může být znak, slovo nebo celá objektová struktura poslaná jako XML struktura [9].

ASP.NET je freeware webový framework pro vývoj webových aplikací. Je nástupcem ASP. Je založen na CLR, který je sdílen všemi aplikacemi postavenými na .NET. Programátoři tak mohou realizovat ASP.NET projekty v jazycích podporující CLR (Visual Basic, C#, Managed C++). ASP.NET projekty je nutno po úpravě přeložit překladačem a následně spustit [10].

Proxy je prostředník mezi cílovým serverem a klientem. Proxy překládá klientské požadavky a vystupuje proti serveru jako klient. Dostane-li odpověď od serveru, pošle ji klientovi, který si o ni požádal. Proxy se používá i pro možnost rozdělení zátěže mezi více serverů [11].

2.3.2 Strana Klienta

Na straně klienta je požadována vizualizace mapových dat, a proto bude aplikován OpenLayers pro zobrazení dat z Geoserveru. Byla-li tato technologie vybrána pro zobrazení dat, lze použít HTML pro tvorbu webové stránky. Kaskádové styly zabezpečují styl webového rozhraní a pro animace, kontrolu a dynamičnost webových stránek je použito Jquery. Technologie AJAX je zde aplikována při přihlašování a registraci uživatele, zobrazení dat z výpočtu hydrogramu a dalších dotazů, bez nutnosti opětovného načtení celé stránky.

HTML je značkovací jazyk pro tvorbu webových stránek. Dokument obsahuje většinou tři části - Doctype, Head, Body. Doctype určuje, o jakou verzi HTML se jedná. Část Head obsahuje metadata stránky, odkazy na CSS a JS soubory, názvy stránky, obrázek stránky a další. A poslední část Body obsahuje samotnou stránku ve značkách párových či nepárových, ze kterých se vygeneruje a zobrazí požadovaná stránka. V této sekci mohou být nadpisy, odstavce, tabulky, obrázky, seznamy a další [12].

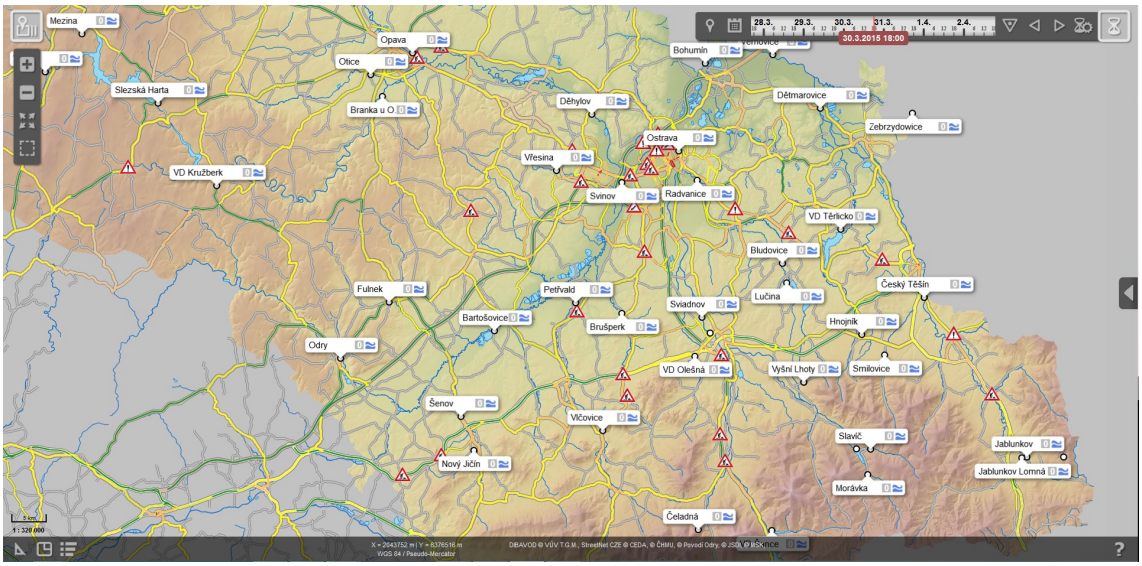
CSS jsou kaskádové styly pro stylování, pozicování a animace HTML elementů. Styl je nastavení vlastností (barva, typ písma, pozicování a dalších) pro element nebo skupiny elementů. Styly pro jednotlivé elementy mohou být uloženy v css souboru a pak připojeny do HTML. Styl uložený v css souboru slouží k oddělení vzhledu od obsahu nebo styl může být přímo u elementu uložen. Zde se kromě stylování dají vytvořit různé animace a transformace elementu. Zmíňovaná funkcionalita se objevila až ve verzi 3 [13].

OpenLayers je opensourcová JS knihovna pro zobrazení mapových dat ve většině moderních webových prohlížečů. Umí reprezentovat vektorové a rastrové vrstvy v mapě. Umožňuje při práci s mapou používat pohyb, přibližování, oddalování, uložení snímku obrazovky a jiné funkce [14].

Jquery je rychlá, malá JS knihovna s podporou většiny prohlížečů. Zjednodušuje syntaxi JS a manipulaci s DOM, ve kterém jsou všechny elementy stránky uloženy. Mezi její další funkce patří výběr DOM elementů pomocí selektorů, zpracování událostí, manipu-

lace s CSS. Jquery také rozšiřuje a zjednodušuje práci s AJAX a přidává nové grafické prvky, efekty a animace [15].

Při použití AJAX neboli asynchronní JS a XML se nejedná o novou technologii, ale o spojení obou dříve uvedených technologií. AJAX je JS knihovna, která umožňuje měnit obsah části webové stránky bez nutnosti načítat celé webové stránky znovu. Po vyvolání události AJAX pošle data na server, server mu vrátí odpověď a AJAX přepíše obsah jen těch elementů, které se změnily. Data mezi serverem a webovou stránkou jsou přenášena ve formátu JSON nebo XML [16]. Ukázka vzhledu uživatelského rozhraní systému Floreon+ je zobrazena na obrázku 1.

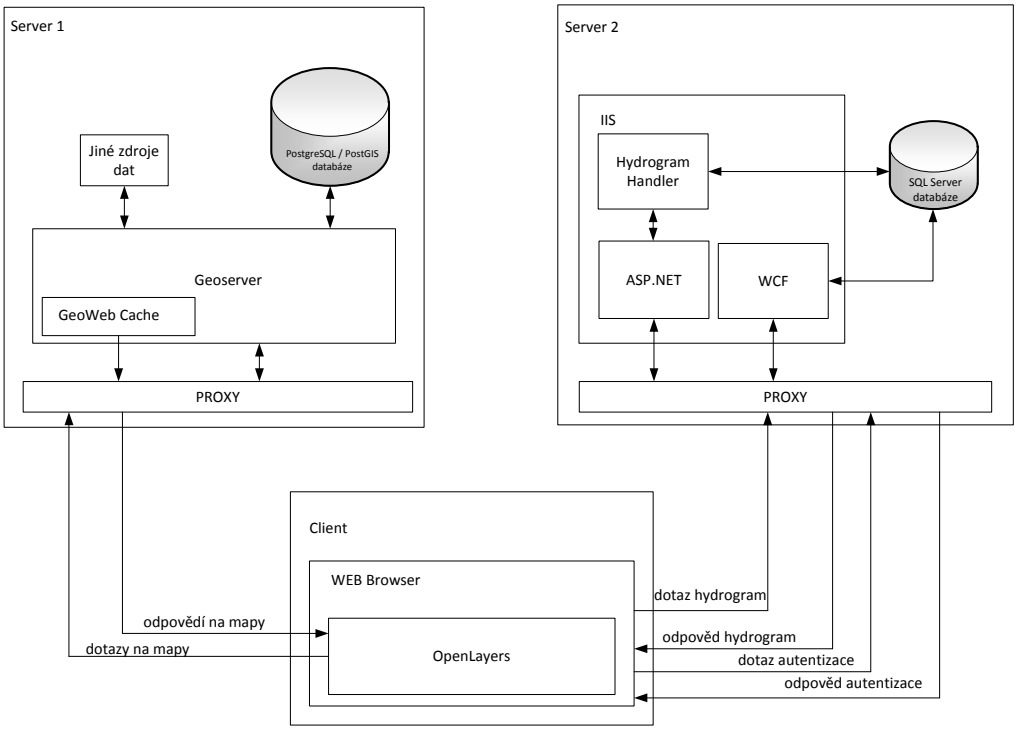


Obrázek 1: Webové rozhraní systému Floreon+

2.4 Funkcionalita systému Floreon+

Pro webové rozhraní systému Floreon+ byly použity dva servery. Na prvém serveru byl nasazen Geoserver a prostorová databáze PostgreSQL. Geoserver zpracovává požadavky klienta z OpenLayers a zpracované jsou zasílány zpět klientovi. O zobrazení mapových služeb WMS a WFS z Geoserveru se stará OpenLayers, díky kterému je možné vypínat a zapínat vrstvy mapy, pohybovat, přibližovat, oddalovat a hlavně počítat vzdálenosti nebo rozlohu nad mapovými vrstvami. Podkladové, říční a silniční mapové vrstvy jsou na webu reprezentovány službou WMS. Vrstvy dopravního toku, simulovaných záplav a dopravních informací používají, pro vykreslení vektorová data, která se berou ze služeb WFS. Zobrazení vrstev závisí na právech přihlášeného uživatele, ne každý uživatel může vidět všechno. O práva se stará Geoserver dle nastavené role uživatele. Na druhém serveru je nasazeno IIS (Internetová informační služba), na kterém běží WCF služba, která je použita pro přihlašování uživatelů, registraci nových uživatelů a k uchovávání informací

o daném uživateli. Seznam uživatelů a jejich data jsou uložena v MS SQL databázi. O simulaci modelovaných oblastí krizového řízení včetně vizualizace jejich výstupu se stará systém naimplementovaný v C#, který s webovým rozhraním komunikuje pomocí webových služeb a ASP.NET knihoven. Diagram rozvržení systému Floreon+ naleznete na obrázku 2.



Obrázek 2: Rozvržení systému

3 Zátěžové testování a testovací plán

3.1 Historie testování

První historicky nalezená počítačová chyba byla objevena v roce 1945 na Harvardské univerzitě, na reléovém počítači, když můra zkratovala kontakty relé. Od této doby mluvíme v informatice o hledání chyb. V 70. létech bylo testování opomíjeno a mezi softwarovými profesionály bylo označováno za druhořadou práci. V tomto období údržba systému stála více než jeho vývoj. Ale během času poptávka po kvalitních testerech rostla a chápání testování také. Nyní testování utváří nezbytnou aktivitu v softwarovém inženýrství a profesionální testeři jsou respektováni jako profesionální vývojáři [17].

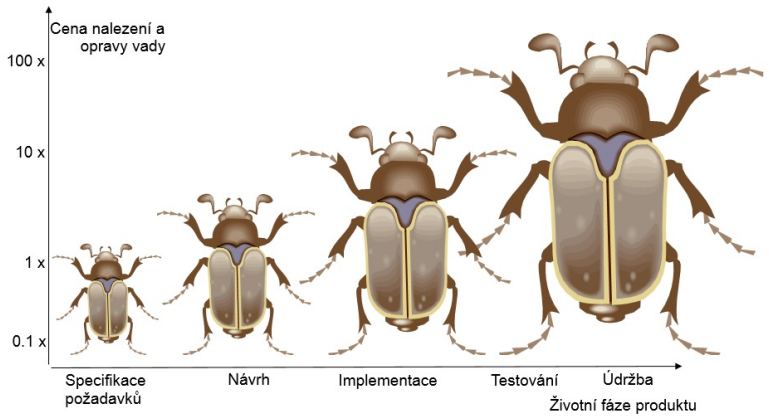
3.2 Testování

Testování neznamená pouze verifikaci běžícího programu, zahrnuje rovněž testování požadavků, revize dokumentace, inspekce kódu, statické analýzy. Testování také zahrnuje nástroje pro testování požadavků, spouštění testů, zátěžové testování a ladění. Používání těchto nástrojů není jednoduché, vyžaduje analytické a programátorské dovednosti. Testování není nudná rutinní činnost, ale jedná se o náročnou práci, která zahrnuje analýzu požadavků, návrh testovacích případů, testovacích plánů a scénářů, programování testovacích skriptů, tak i zhodnocení testů a následnou implementaci vylepšení a opětovné otestování. Během testování nesmíme uvažovat, že program je bez chyb, byli bychom ovlivněni. Proto je vhodné testovat na datech, u nichž je malá pravděpodobnost zadání. Systém nelze nikdy otestovat celý, ale pouze jen z větší části. Přitom se však některé chyby nemusí zobrazit ani při testování, ale až při zpracovávání programu, ale díky testování je počet těchto chyb redukován na minimum [18].

Testování systému je žádoucí provádět v průběhu vývoje systému a nejen na konci ve fázi dohotovení projektu. Náklady na opravu nalezené chyby v průběhu vývoje software jsou nižší než oprava nalezené chyby při závěrečném testování. Na obrázku 3 jsou znázorněny náklady na nalezení a opravu chyby během vývoje softwaru [18].

3.3 Terminologie použitá v testování

- Chyba (Error) - stav systému, kdy systém není schopen poskytovat služby v požadované kvalitě a vede k selhání systému, může se jednat o lidský, procesní nebo strojový omyl [17]
- Vada (Fault) - chybná část kódu, která je následkem pochybení člověka, a tedy příčinou chyby [17]
- Selhání (Failure) - nesoulad mezi aktuálním a specifikovaným chováním systému (stane se provedením vadného kódu [17])



Obrázek 3: Náklady na opravu chyb v průběhu času [18]

3.4 Testovací dokumentace

Testovací dokumentaci lze rozdělit do čtyř částí - zásady testování, strategie tetování, hlavní plán testování a plán pro jednotlivé úrovně testování. Návaznost a propojení jednotlivých částí je zřejmá z obrázku 4.

- Zásady testování popisují hlavní cíle a celkový postup, jakým daná organizace obecně k testování přistupuje.
- Strategie testování se zabývá požadavky a způsoby, jakými je testování prováděno, přestože stále zachovává dostatečnou míru abstrakce, aby ji bylo možno použít u většiny projektů.
- Hlavní plán testování je již specifické zaměření na konkrétní projekt a popisuje, jak bude v dané situaci konkrétně uplatněna strategie testování.
- Plán pro jednotlivé úrovně testování je použit zejména pro rozsáhlejší projekty. Rozšiřuje hlavní plán o plány jednotkového testování, plány integračního testování a podobně.

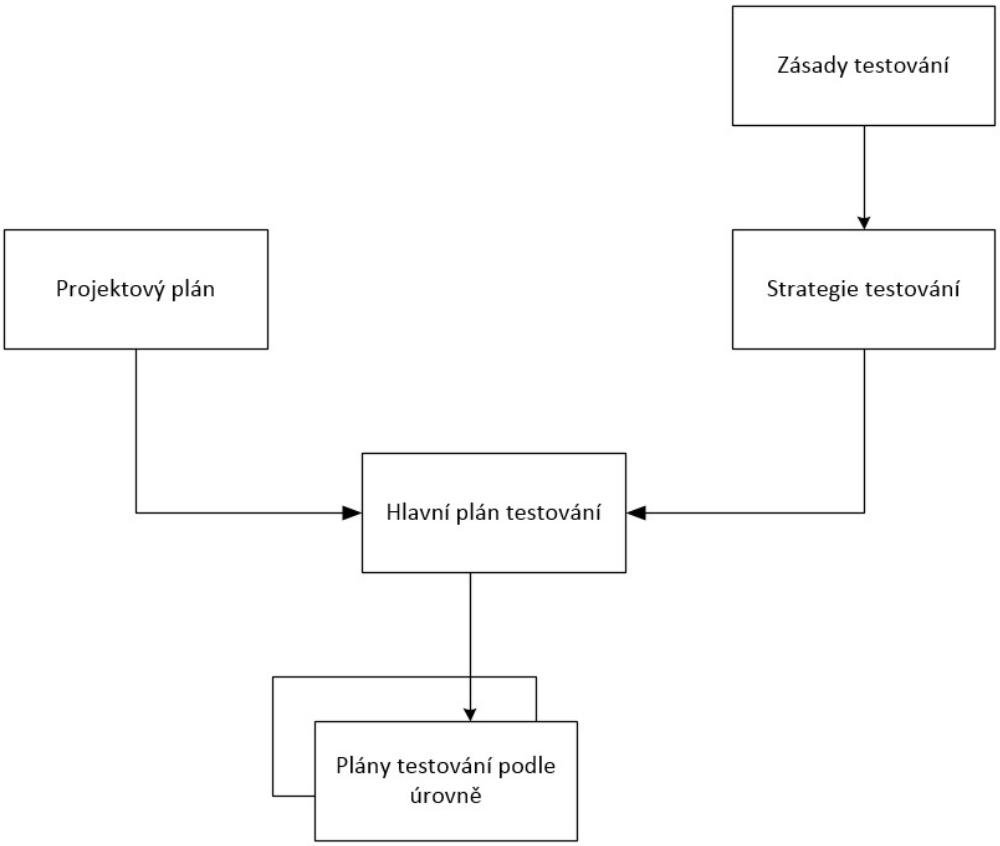
3.5 Plán testování

Plán testování je dokument, který vymezuje:

- Cíl testování – s jakým cílem je testování prováděno. Zda jde o zátěžové testování nebo o otestování funkčnosti systému
- Rámec testování – co bude předmětem testování
- Způsob testování, strategie – jaké techniky, nástroje budou použity

- Potřebné zdroje – hardware, lidé
- Identifikace kritérií pro přerušení testování – za jakých podmínek skončí
- Rizika ohrožující testování – rozpočet, zdroje

Plán testování lze tedy chápat jako specifickou aplikaci strategie a organizaci procesů testování, a musí být v souladu s projektovým plánem. Slouží nejen vývojářům, manažerům a testerům k zajišťování kvality, ale může být přístupný i zákazníkovi nebo auditu. K plánování testů dochází už vrámci specifikace požadavků, kde výstupní kritéria budou představovat podmínky pro finální přijetí projektu. Testovací plán se většinou mění během vývoje či testování, z důvodu zjištění informací během vývoje software [17].



Obrázek 4: Plánování testování

3.6 Chování webové stránky

Od webové stránky se vyžaduje hlavně dostupnost, rychlost a spolehlivost. Každého uživatele většinou dlouhá přístupová doba nebo nedostupnost stránky odradí, proto se kvůli náporu uživatelů webové stránky zátěžově testují. V běžných podmínkách není možné zajistit, aby web testovalo tisíce uživatelů, testování se tedy většinou provádí pomocí testovacích plánů napsaných v testovacích nástrojích. Zátěž zde vytváří počítačový program, který ušetří finance a je dostupnější než zapojení tisíců lidí. Zátěžové testy se tvoří pro konkrétní skupinu stránek. Například stránky pro práci s mapou nebudou testovány stejným způsobem jako reprezentační stránky firmy.

V případě webového rozhraní systému Floreon+ je vyžadována co nejkratší odezva a vysoká dostupnost i pro současný přístup desítek až stovek uživatelů. Proto jsem v testech sledoval průměrný čas odezvy a počet chyb pro danou kategorii dotazů.

3.7 Zátěžové testování

Cílem zátěžového testování není jen najít chyby, ale i odstranit překážky, které brání určené uživatelské zátěži. Stanovuje dobu, za jakou se systém znovu zotaví. Jedná se o destruktivní testování. V průběhu testů se počet uživatelů neustále zvyšuje, dokud aplikace nespadne. Při testování je dobré sledovat jak stavy hardwarových prostředků, např. stav procesoru, paměti a dalších, tak i softwarové logovací soubory programů na serveru. Toto sledování nám může pomoci s případnou optimalizací v případě selhání. Většinou se používá black-box přístup při spouštění testů. Jedná se o přístup, kdy testeři nevědí nic o aplikaci a zdrojových kódech, pouze testují aplikaci nebo její modul jako uzavřený celek dostupný jen přes své rozhraní. Jestliže nějaká část systému nesplňuje požadavky, je nutno ji optimalizovat a zátěžový test nad rozhraním vykonat znovu. Tento postup se opakuje, dokud není zaručena požadovaná kvalita systému [19].

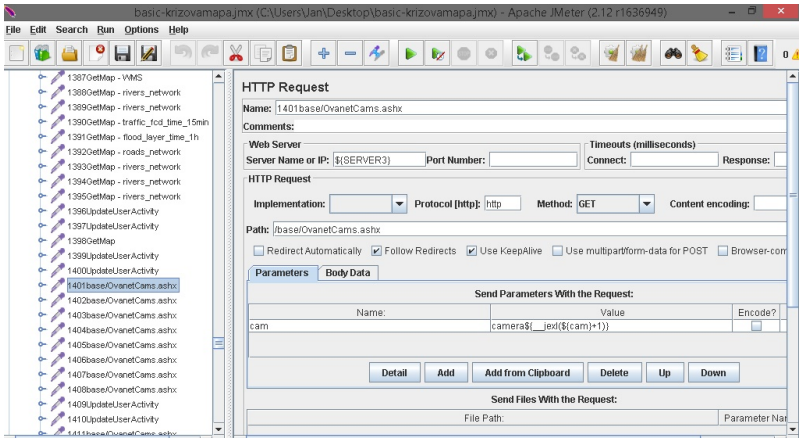
3.8 Testovací nástroje

Pro testování výkonu webové aplikace jsem zvolil nástroje JMeter a testovací platformu Visual Studia. Tyto nástroje umožňují vydefinovat a provést black-box testy webových aplikací tím, že generují a posílají vydefinované požadavky na server webové aplikace, analyzují odpovědi a měří jejich odezvu. Jestliže testovací nástroje berou 100% výkonu počítače, existuje zde reálná možnost zkreslení výsledků testů. Zkreslení výsledků omezíme rozložením testů na více počítačů použitím distribuovaného testu, kdy obdržíme přesné hodnoty o stavu a rychlosti odezvy testovaného systému.

3.8.1 JMeter

JMeter je opensourcová aplikace naprogramovaná v Javě, navržená pro zátěžové testování funkcionality, chování a měření výsledků. Původně byla navržena pro testování webových rozhraní, ale později byla rozšířená o další testy. JMeter umí testovat HTTP,

HTTPS webové protokoly, dále také SOAP/REST protokoly, a také protokoly pro poštovní klienty (POP3, SMTP) a souborové protokoly jako FTP. Aplikace umožňuje přidávat různé pluginy pro ulehčení práce [20]. Ukázka testovacího plánu v programu JMeter je znážorněna z obrázku 5.



Obrázek 5: JMeter - ukázka testovacího plánu

3.8.2 Visual Studio

Visual Studio je aplikace od společnosti Microsoft určena pro vývoj a testování vyvíjeného programu nebo webového rozhraní. Původně byla určena jen pro vývoj a ladění v programovacích jazycích určených pro desktopové aplikace, později byla dopracovaná podpora vývoje webových aplikací. Byly také přidány zátěžové testy, které jsou dostupné až ve verzi Ultimate. Ukázka webtestu je zobrazená na obrázku 6. Visual Studio poskytuje testování buď z lokálního počítače, nebo z webového cloudu pomocí online účtu, když výkon počítače už není dostačující [21].

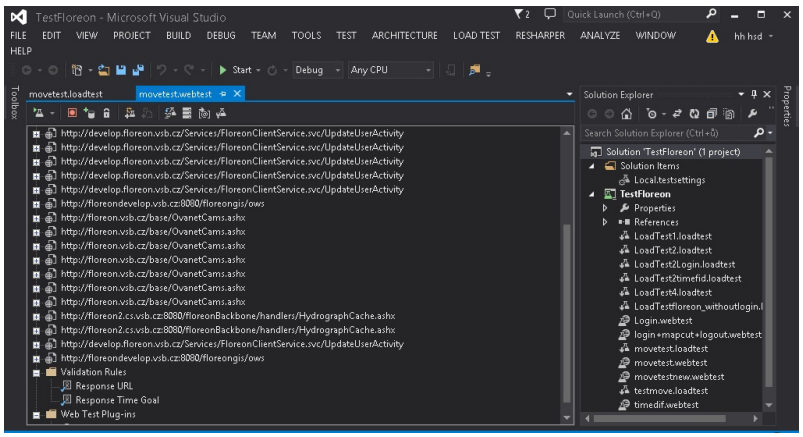
3.9 Testovací plán pro webové rozhraní Floreon+

Nad webovým rozhraním Floreon+ byl vytvořen hlavní testovací plán, který obsahuje tři testovací plány ve dvou testovacích nástrojích (v JMeter, Visual Studiu), které mají za úkol otestovat celé webové rozhraní a vyexportovat data do textového dokumentu jak z hlediska neplatných žádostí na server, tak i z hlediska zátěžových testů.

3.9.1 Základní testovací plán

Tento testovací plán byl vytvořen z důvodu otestování chování mapových vrstev a zobrazení hydrogramů. Test obsahuje nejzákladnější testování webového rozhraní jako je jeho chování při načítání mapových vrstev do klientova prohlížeče, práce s mapovými vrstvami - přiblížení, oddálení, zobrazení konkrétního zobrazení mapy. Testovaní plán

dále zahrnuje zobrazení hydrogramu k měřící stanici na vodním toku. Ve vývojovém nástroji JMeter test zahrnuje kolem 1800 žádostí na server. Testovací plán obsahuje náhodné hodnoty na parametry hydrogramu a kamer.



Obrázek 6: Visual Studio - ukázka webtestu

3.9.2 Login testovací plán

Vytvoření testovacího plánu slouží k otestování rychlosti WCF služby, která zajišťuje přihlašování a registraci uživatelů do systému. Dále testovací plán zhodnotil vytváření mapových výřezů mapy a jejich uložení u uživatele a také změnu nastavení uživatelského účtu. Ve vývojovém nástroji JMeter testovací plán zahrnuje kolem 600 žádostí na server.

3.9.3 Testovací plán Posuv času

Testovací plán měl za úkol otestovat webové rozhraní systému v měnícím se čase a rychlost načítání hodnot povodňových stupňů a mapových vrstev při změně času. Ve vývojovém nástroji JMeter testovací plán zahrnuje kolem 350 žádostí na server.

3.10 Popis testování

Jelikož je testované webové rozhraní naimplementované výhradně pomocí technologií klientského skriptování, je celá zátěž spojená s vizualizací přenesena na klientův prohlížeč. Tato část neprovádí žádné náročné výpočty a jen zobrazuje data získaná ze serveru. Hlavní částí mé bakalářské práce byla proto optimalizace serverové části systému Floreon+ a jeho odolnost vůči vysoké zátěži. Z tohoto důvodu byly vytvořeny testovací plány v aplikacích JMeter a VisualStudio. Po vytvoření testovacích plánů byl systém podroben každému testovacímu plánu pro zátěž padesáti uživatelů. V následujících tabulkách jsou zobrazeny výsledky testů na produkční verzi systému, jejíž nasazení se řídilo diagramem 2.

Název dotazů	Průměrná doba odezvy pro skupinu uživatelů [ms]				
	$1-10$	$11-20$	$21-30$	$31-40$	$41-50$
$GetMap$	303	358	522	734	611
$GetMap-flood\_layer$	182	178	184	181	187
$GetMap-Minimap$	518	714	752	696	865
$GetMap-rivers\_network$	750	706	713	780	701
$GetMap-roads\_network$	1320	1568	1551	1396	1391
$GetMap-traffic\_fcd$	216	201	212	203	200
$GetMap-WMS$	260	276	291	300	300
$GetNotAvailableLayers$	44	47	44	95	43
$GetStationSPAsJSON$	1147	1581	1833	1860	1638
$Hydrogram1$	1744	3480	4509	5958	7416

Tabulka 1: Výsledky testu Posuv času z JMeter - bez úprav rozhraní

Název dotazů	Průměrná doba odezvy pro skupinu uživatelů [ms]				
	$1-10$	$11-20$	$21-30$	$31-40$	$41-50$
$GetBaseMap$	464	235	664	1107	269
$GetDataFromCokies$	392	76	46	46	40
$GetMap$	359	548	519	523	536
$GetMap-flood\_layer$	279	297	326	313	318
$GetMap-Minimap$	636	919	974	932	1052
$GetMap-rivers\_network$	676	705	738	760	754
$GetMap-roads\_network$	803	829	880	908	834
$GetMap-traffic\_fcd$	315	306	327	320	321
$GetMap-WMS$	268	254	276	298	265
$GetNotAvailableLayers$	37	39	38	37	37
$GetStationSPAsJSON$	2317	2935	3472	5089	5030
$GetUserSettingsList$	23	26	28	24	22
$UpdateUserActivity$	23	23	28	27	23
$Hydrogram1$	2540	2401	2360	2390	2284

Tabulka 2: Výsledky Základního testu z JMeter - bez úprav rozhraní

Z tabulek 1 a 2, kde jsou zobrazeny průměrné hodnoty v milisekundách pro skupinu deseti uživatelů na typ dotazu ze základního testovacího plánu a Posuv času tesovacího plánu v JMeter. Z výsledků je zřejmé, že největší zpoždění testů způsobil výpočet hydrogramu a GetStationSPAsJSON dotazy. Důsledkem zpoždění je, že při každém dotazu musí aplikace sáhnout do relační databáze. V případě GetStationSPAsJSON jsou data z databáze vrácena ve formátu JSON, ale v případě dotazu na hydrogram je z dat vykreslen hydrogram, který je pak poslán do webového rozhraní. Požadavek GetStationSpasJSON natahuje data do rozhraní pro měřící stanice na říčních tocích, kde je později v mapě zobrazen. U výsledků z testu Posuv času je patrné, že požadavek na hydrogram je zpracován, ale při tomto testu se hydrogram vůbec nezobrazuje. V tomto případě zá-

těžový test poukázal na chybu ve funkčnosti aplikace a neplatné volání bylo z aplikace odstraněno. Grafy k tabulkám jsou uvedeny v přílohách10, 11.

Název dotazů	Průměrná doba odezvy pro skupinu uživatelů [ms]				
	$1-10$	$11-20$	$21-30$	$31-40$	$41-50$
$GetMap$	327	302	508	1030	1166
$GetMap-flood\_layer$	261	258	269	277	272
$GetMap-Minimap$	574	746	570	589	772
$GetMap-rivers\_network$	675	652	649	631	641
$GetMap-roads\_network$	867	842	878	878	872
$GetMap-traffic\_fcd$	427	438	427	434	423
$GetMap-WMS$	257	260	281	263	268
$GetNotAvailableLayers$	46	47	46	47	47
$GetStationSPAsJSON$	1137	1005	945	961	926
$GetUserSettingsList$	25	27	26	24	26
$Hydrogram1$	1582	2865	4050	5308	7761
$LoginToSystem$	23	23	22	23	22
$LogoutFromSystem$	20	20	20	20	20
$SetUserSettingsList$	20	20	20	29	20
$UpdateUserActivity$	21	21	21	21	20

Tabulka 3: Výsledky Login testu z JMeter - bez úprav rozhraní

Tabulka 3 je věnována testování služeb WCF, které na systému Floreon+ slouží pro registraci, přihlášení a uchovávání dat. Tento test opět ukázal na neplatné zobrazení hydrogramu, které se při testovaných funkcích nemělo provádět. Dále testy ukázaly, že rychlost WCF je pro testovaný systém více než dostačující. Testy v programu JMeter prověřily dobu odezvy Geoserveru i dobu odezvy jednotlivých mapových vrstev. Graf k tabulce je zobrazen v příloze 12.

Název dotazů	$min[ms]$	$max[ms]$	$avg[ms]$
$GetStationSPAsJSON$	850	1006	930
$Geoserver$	140	230	170
$GoogleAPI-AuthenticationService$	54	190	82
$GoogleAPI$	49	1114	240
$GetNotAvailableLayers$	70	110	80
$UpdateUserActivity$	39	44	41
$SetUserSettingsList$	39	45	41
$LoginToSystem$	43	60	46
$LogoutFromSystem$	38	59	41
$GetUserSettingsList$	39	45	41

Tabulka 4: Výsledky Login testu z Visual Studia - bez úprav rozhraní

Název dotazů	$min[ms]$	$max[ms]$	$avg[ms]$
$Hydrogram1$	4170	23300	14000
$OvanetCams$	1320	21800	7320
$GetStationSPAsJSON$	870	4980	1300
$Geoserver$	150	240	180
$GoogleAPI - AuthenticationService$	140	180	150
$GoogleAPI$	100	500	290
$GetNotAvailableLayers$	74	87	77
$UpdateUserActivity$	41	52	45
$GetUserSettingsList$	41	63	52

Tabulka 5: Výsledky Základního testu z Visual Studia - bez úprav rozhraní

Název dotazů	$min[ms]$	$max[ms]$	$avg[ms]$
$Hydrogram1$	7930	58200	38600
$GetStationSPAsJSON$	900	13200	5430
$Geoserver$	130	290	210
$GoogleAPI - AuthenticationService$	55	190	110
$GoogleAPI$	49	110	84
$GetNotAvailableLayers$	74	95	77
$UpdateUserActivity$	41	65	47
$GetUserSettingsList$	39	58	44

Tabulka 6: Výsledky Posuv času testu z Visual Studia - bez úprav rozhraní

Po testech v JMeter na produkční verzi byly provedeny testy v testovacím prostředi Visual Studio, kde byly vytvořeny tytéž tři testovací plány jako webtesty. Webtesty poté sloužily jako podpůrné testy pro Load testy na určitý počet uživatelů. U testu už nejsou uvedeny časy pro určité uživatelé, ale pouze minimální, průměrné a maximální časy pro počet uživatelů, na který byl test nastaven. Výsledky testů z programu Visual Studio v tabulkách 4, 5, 6 (grafy: 15,13, 14) ukazují, že tento testovací nástroj není schopen otestovat mapové vrstvy, ale jen odezvu Geoserveru. Při požadavcích testy z programu Visual Studio identifikovaly, co program JMeter neukázal - volání skriptů od společnosti Google, které jsou na webovém rozhraní použity. Z výsledků je zřejmé, že jednotlivé nástroje testují webové rozhraní odlišným způsobem. V každém testovacím nástroji se objevily odlišné požadavky při stejných krocích nad webovým rozhraním, proto by měl být nad webovým rozhraním proveden vždy test ve více testovacích prostředích.

4 Úpravy a optimalizace webového rozhraní

Na základě výsledků testů na webovém rozhraní systému Floreon+ byly s cílem zvýšení výkonu při vysokém počtu uživatelů zvoleny nutné úpravy. Požadavky na OvanetCams nebyly optimalizovány, protože data byla dodána jiným poskytovatelem. Z výsledků testů je patrné, že v testech před úpravou webového rozhraní systému Floreon+ byly kamery nedostupné (dotazy na ně se nezobrazily v tabulkách), ale v testech po optimalizaci už dostupné byly a mohl být otestován a změřen čas odezvy dotazů na kamery.

4.1 Přenesení hydrogramu na jiný server

Z produkční verze, kde byl testován dříve, byl hydrogram přenesen na testovací server. Testovací server měl sice nižší hardwarový výkon, jenž byl dedikován jen na výpočet hydrogramu. Na všech serverech byly prováděny základní testovací plány, poněvadž ostatní testovací plány zahrnovaly neplatné dotazy na hydrogram, proto je nebylo možné použít. První test zahrnoval test hydrogramu bez přenesení databáze, tzn. že databáze s daty pro výpočet hydrogramu se nachází na stejném serveru jako program pro výpočet hydrogramu. Druhý test hydrogramu byl nad přenesenou databází a třetí test byl spuštěn na přenesené databázi a jiném serveru se stejným výkonem jako produkční server. Výsledky této optimalizace zobrazuje tabulka 7.

Název dotazů	průměrný čas v ms
Produkční server	2395
Testovací server bez přenesené databáze	33404
Testovací server s přenesenou databází	29913
Testovací server 2 s přenesenou databází	2927

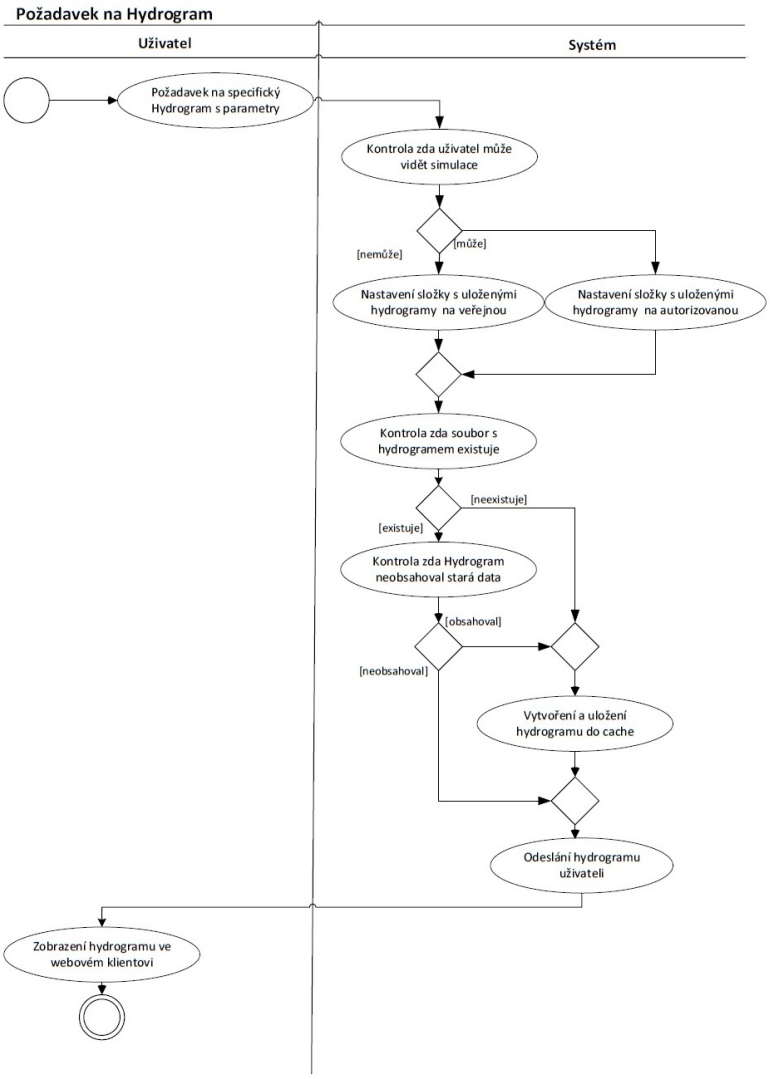
Tabulka 7: Závislost výkonu serveru na výpočtu hydrogramu

Z výsledků v tabulce 7 plyne, že výpočet hydrogramu je náročný proces. Slabšímu testovacímu serveru, který měl pouze tři jádra procesoru, scházel výkon, přestože na serveru běžel jen výpočet hydrogramu. Dále výsledky ukazují, že doba odezvy na druhém testovacím serveru byla srovnatelná s odezvou na produkčním serveru. Proto testy na hydrogram a úpravy hydrogramu budou probíhat na testovacím serveru 2, abychom zabránili shození produkčního serveru.

4.2 Cachovaní hydrogramu

Z důvodu vysoké hardwarové náročnosti při volání hydrogramu (výsledku simulace) pro stanici na určitém vodním toku byl vytvořen cachovací algoritmus. Aplikace pro vrácení a vykreslení hydrogramu z dat byla napsána v ASP.NET, proto byl algoritmus vytvořen v handleru. U algoritmu cachování hydrogramu je restrikce, kde se ověřuje dle autorizačního kódu uživatele, zda může nebo nemůže vidět výpočet budoucího stavu pro stanici. Podle této restrikce se pošle hydrogram s budoucím stavem nebo jen s aktuálním. Algoritmus při prvním požadavku zjistí, zda hydrogram není uložen už v cachi.

Pokud je uložen, vrátí klientovi obrázek pro dané parametry dotazu. Pokud ale není vytvořen, pošle dotaz na vytvoření hydrogramu, ten se následně uloží a je už cachovaný. Pokud dotaz na hydrogram proběhl jako možná budoucí předpověď, bude hydrogram odstraněn z cache a vytvořen znovu, protože data byla vypočítána z předpovědi, a tudíž se změnila. Z pohledu rychlosti vyplývá, že pokud hydrogram není v cachi (první dotaz), je algoritmus stejně rychlý jako algoritmus bez cache, ale v případě druhého dotazu na stejný hydrogram už je dotaz mnohem rychlejší. Může nastat situace, že jsou pro stejnou stanici uloženy a vypočteny dva hydrogramy, jeden pro autorizovaného uživatele s budoucím stavem a druhý jen s akutálním stavem, budou hydrogramy uložený v rozdílných složkách. Diagram cachování hydrogramu je zobrazen na obrázku 7.



Obrázek 7: Aktivitní diagram cachování hydrogramu

4.3 Cachovaní SPA

Z výsledků testů bylo zjištěno, že obdržení odpovědi ohledně stavu stupně povodňové aktivity pro daný čas trvá docela dlouho. Toto zpoždění je způsobeno složitým výběrem dat z tabulek, porovnáváním s konkrétní tabulkou, kde jsou uloženy stavy pro měřící stanice, následná konverze do JSON formátu a zaslání odpovědi na dotaz klientovi. Protože je používán JSON formát, lze data lehce ukládat, a tak je cachovat. Byl zde použit podobný cachovací algoritmus jako v případě cachování hydrogramu, kdy přijde na server dotaz na data starší než aktuální, uloží se data permanentně do cache. Pokud se dotazujeme na aktuální stav, tak platnost cachovaného obsahu se po čase vytrácí, což je zapříčiněno tím, že data pro budoucnost jsou proměnlivá. Data budou permanentně uložena jen v případě jejich neměnnosti. Ukázka části zdrojového kódu cachovacího algoritmu je uvedena ve výpisu 1.

```
public void ProcessRequest(HttpContext context) {
    string rdatetime = context.Request.QueryString["datetime"];
    string authenticationKey = context.Request.QueryString["authkey"];
    string pathrequest = context.Request.QueryString.ToString();
    string filepath = CreateFilePath(context, authenticationKey, rdatetime);
    try {
        if ( File.Exists( filepath ) ) {
            if ( IsCachedResultExpired(filepath, Convert.ToDateTime(rdatetime)) )
                CreateAndSaveResult(context, pathrequest, filepath);
            SendResult(context, filepath);
        }
        else {
            CreateAndSaveResult(context, pathrequest, filepath);
            SendResult(context, filepath);
        }
    }
    catch ( Exception e ) {
        log.Error("Could not send SPA on " + filepath, e);
        context.Response.ContentType = "text/plain";
        context.Response.Write("There was an error while loading the flood warning
            levels.");
    }
}
```

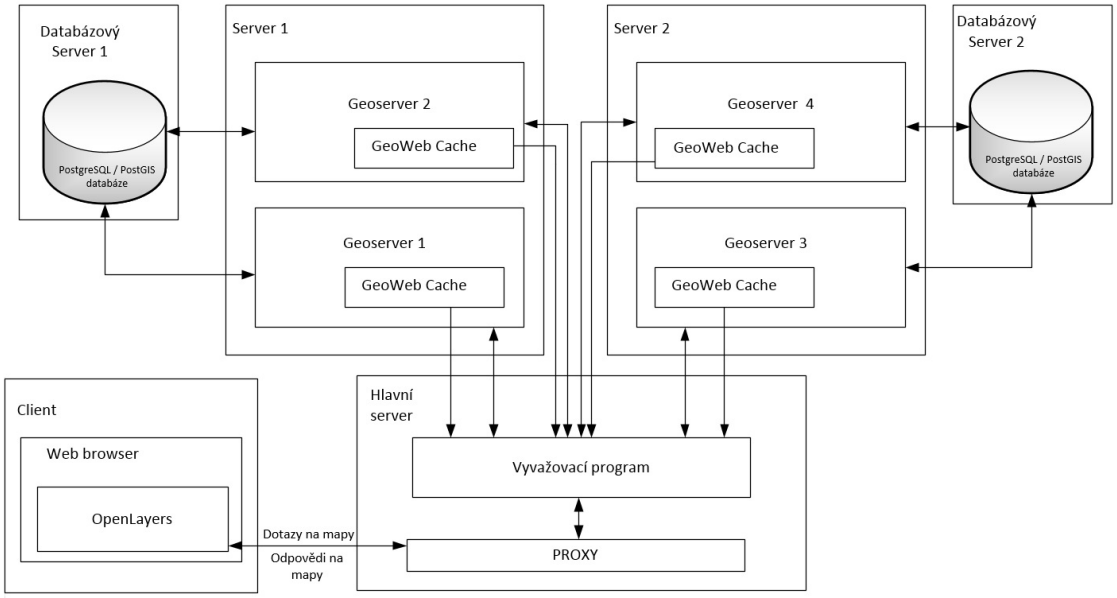
Výpis 1: Funkce pro uložení a zaslání odpovědi cachování SPA

4.4 Rozvržení systému

V základním rozvržení na obrázku 2 byla jen jedna instance Geoserveru a systém byl plánován na zatížení stovkami uživatelů. Instance Geoserveru by tak velký počet uživatelů neustála. U dotazu na mapové vrstvy systém vykazoval vysoké zatížení hardwarových prostředků již při zatížení padesáti uživateli. Na základě tohoto zjištění bylo vypracováno nové rozvržení systému pro lepší zvládání očekávaného většího zatížení systému. Proto byly vytvořeny čtyři instance Geoserveru, kde dvě instance Geoserveru

byly umístěny na jeden ze dvou serverů, kde běží operační systém Linux. Těmto Geoserverům dodávají data dvě PostgreSQL databáze, přičemž druhá databáze je replikou první. Každá z databází běží na databázovém serveru přístupném jen z lokální sítě. Celá infrastruktura je skryta za proxy serverem, který zajišťuje nejen bezpečnost, ale i rozdělování zátěže. Server s IIS, na kterém je nasazen program na výpočet hydrogramu a WCF služba, byl zachován, ale pro zjednodušení, viz obrázek 8, se v diagramu nenachází.

Pro výběr serveru zpracovávajícího dotazy klientů, byl vytvořen speciální vyvažovací program, ten v případě vytížení jedné instance z Geoserveru pošle dotazy na další instanci Geoserveru. Díky navrženému řešení byl navýšen počet uživatelů systému a snížila se přístupová doba k mapovým vrstvám pro stávající počet uživatelů.



Obrázek 8: Rozvržení systému po úpravách

4.5 Úprava webového klienta

Pro správnou a rychlejší funkčnost webového klienta musely být provedeny úpravy webového rozhraní systému Floreon+.

4.5.1 Lokální umístění skriptů

Z výsledků testů programu Visual Studio je zřejmé, že rozhraní během testů volalo Google scripty. Část skriptů, které byly volány, byla umístěna lokálně, zbylá část zůstala na internetu. Nyní není nutné stahovat JS scripty z webu, neboť jsou umístěny lokálně a volají se přímo.

4.5.2 Odstranění nadbytečných požadavků

Vytváření výsledků z testových plánů z programů JMeter a Visual Studio ukázalo, že z webového prohlížeče vycházejí zbytečné požadavky na hydrogram, přičemž hydrogram nebyl nikde načítán, ani zobrazován. Proto byly prozkoumány zdrojové kódy a nalezeny požadavky na výpočet hydrogramu. Nalezené požadavky na hydrogram se ukázaly zcela zbytečné, byly proto odstraněny. Ve výsledcích dalších testů se dotaz na hydrogram už nezobrazoval. Dále bylo zredukováno volání method GetUserSettingList a SetUserSettingList ze služby WCF, jež byly v nastavení uživatelského účtu v některých místech volány zbytečně.

5 Zhodnocení výsledků testů a optimalizací

Tato kapitola obsahuje zobrazení výsledků testů po výsledné optimalizaci systému Floreon+. Z výsledných dat je zřejmý nárůst rychlosti zpracování některých požadavků.

5.1 Výsledky testů po úpravách

Z výsledků testů, uvedených v tabulkách 8, 9, 10 z programu JMeter je patrné, že na rozdíl od předchozího testu byly kamery dostupné a jejich průměrná odezva se pohybovala kolem 820 milisekund. Je evidentní, že první dotaz na hydrogram trval skoro stejnou dobu jako volání hydrogramu před cachovacím algoritmem. V případě druhého téhož dotazu, odezva trvala v průměru 163 milisekund, což přináší v případě vícenásobného opakovaného volání 14,7 x větší zrychlení. U dotazu GetStationSPAsJSON před cachováním dotaz trval 1611 milisekund, po úpravách odezva na nacachovaná data klesla přibližně na 85 milisekund. Tato úprava tedy přináší devatenáctinásobné zrychlení. Porovnání zrychlení je uvedeno na obrázku 9. Z výsledků testovacích plánů Login a Posuv času byla eliminována nadbytečná neplatná volání dotazů. Nové rozvržení systému přineslo výrazné změny u náročnějších mapových vrstev např. říční a silniční síť, kde se odezva snížila o čtvrtinu. U ostatních vrstev změny nebyly výrazné. Současně došlo ke snížení chybovosti u odpovědí na dotaz a snížilo se vytížení Geoserveru oproti starému schématu rozvržení systému, v němž byla použita jen jedna instance Geoserveru. Graf k výsledkům testů z JMeter je uveden v přílohách 19, 20, 21.

Název dotazů	Průměrná doba odezvy pro skupinu uživatelů [ms]				
	$1-10$	$11-20$	$21-30$	$31-40$	$41-50$
$base/OvanetCams.ashx$	823	836	826	879	792
$GetBaseMap$	2242	960	818	1514	1949
$GetDataFromCokies$	407	1623	1197	769	373
$GetMap$	586	883	700	774	1111
$GetMap-flood\_layer$	554	576	589	530	556
$GetMap-Minimap$	895	700	672	503	614
$GetMap-rivers\_network$	808	756	739	764	769
$GetMap-roads\_network$	683	692	686	695	693
$GetMap-traffic\_fcd$	594	605	611	570	610
$GetMap-WMS$	827	846	851	834	830
$GetNotAvailableLayers$	73	59	83	43	77
$GetStationSPAsJSON$	785	107	68	441	441
$GetUserSettingsList$	311	217	86	30	917
$UpdateUserActivity$	58	78	65	65	87
Hydrogram 1 před cache	2612	3202	3201	2854	2768
Hydrogram 1 po cache	129	334	89	119	145

Tabulka 8: Výsledky Základního testu z JMeter - po úpravách rozhraní

Název dotazů	Průměrná doba odezvy pro skupinu uživatelů [ms]				
	$1-10$	$11-20$	$21-30$	$31-40$	$41-50$
$GetMap$	285	291	276	283	272
$GetMap-flood\_layer$	213	217	214	221	212
$GetMap-Minimap$	424	351	239	240	256
$GetMap-rivers\_network$	403	409	397	407	397
$GetMap-roads\_network$	328	329	328	326	332
$GetMap-traffic\_fcd$	233	230	230	230	234
$GetMap-WMS$	329	357	367	347	365
$GetNotAvailableLayers$	53	57	56	57	51
$GetStationSPAsJSON$	110	78	47	141	47

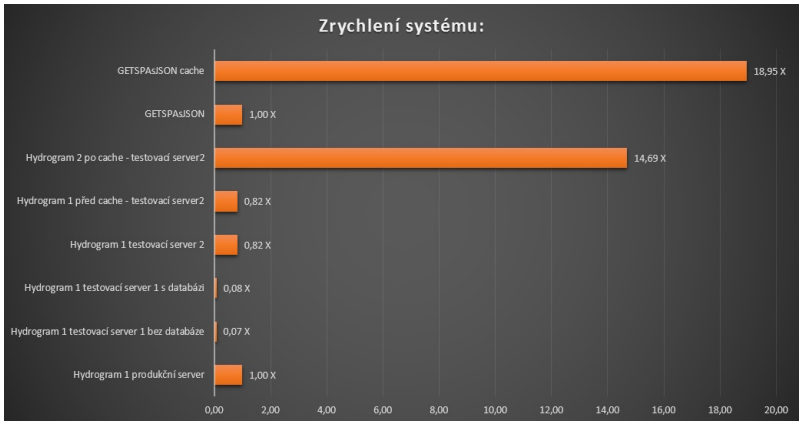
Tabulka 9: Výsledky Posuv času testu z JMeter - po úpravách rozhraní

Název dotazů	Průměrná doba odezvy pro skupinu uživatelů [ms]				
	$1-10$	$11-20$	$21-30$	$31-40$	$41-50$
$GetMap$	2090	838	1444	472	857
$GetMap-flood\_layer$	565	540	494	527	549
$GetMap-Minimap$	742	806	785	917	672
$GetMap-rivers\_network$	772	733	791	767	790
$GetMap-roads\_network$	750	820	725	756	759
$GetMap-traffic\_fcd$	616	677	681	644	673
$GetMap-WMS$	893	917	874	962	927
$GetNotAvailableLayers$	782	63	51	55	65
$GetStationSPAsJSON$	149	1412	706	143	86
$GetUserSettingsList$	55	50	91	61	33
$LoginToSystem$	59	28	30	41	57
$LogoutFromSystem$	28	38	25	37	24
$SetUserSettingsList$	61	45	30	42	70
$UpdateUserActivity$	41	54	47	56	77

Tabulka 10: Výsledky Login testu z JMeter - po úpravách rozhraní

Výsledky testů z programu Visual Studio viz tabulky 11, 12, 13 ukazují , že část Google skriptů byla umístěna z webu do aktuální složky s webovým rozhraním systému Floreon+. V testu umístění skriptů se projevilo odstranění dotazu na Google API. Dále se díky cachovacím mechanismům zrychlil výpočet hydrogramu, přičemž v požadavku hydrogram 1 jsou zahrnuty dotazy, jak na předcachovaný, tak i na nepředcachovaný hydrogram. U dotazu na GetStationSPAsJSON je v testech ve Visual Studio změna nejvýraznější, zrychlení bylo největší. Přesto je patrný nárůst hodnot u WCF dotazů, což bylo důsledkem toho, že cachovací mechanismus GetStationSPAsJSON byl umístěn na stejný server jako WCF služba a jeho výpočetní náročnost zvyšovala vytížení serveru a snižovala volné prostředky WCF službě. U testovacího plánu Login a Posuv času, byly odstraněny neplatné volání hydrogramu a část nadbytečných volání WCF autentizace

byla eliminována. K testům z Visual Studia jsou grafy v přílohách 16, 17, 18.



Obrázek 9: Zrychlení systému

Název dotazů	$min[ms]$	$max[ms]$	$avg[ms]$
$Hydrogram1$	260	5630	1230
$OvanetCams$	720	800	760
$GetStationSPAsJSON$	61	1920	670
$Geoserver$	110	3400	730
$GoogleAPI - AuthenticationService$	140	220	140
$GetNotAvailableLayers$	74	110	85
$UpdateUserActivity$	40	74	45
$GetUserSettingsList$	40	580	220

Tabulka 11: Výsledky Základního testu z Visual Studia - po úpravách rozhraní

Název dotazů	$min[ms]$	$max[ms]$	$avg[ms]$
$GetStationSPAsJSON$	30	3250	81
$Geoserver$	130	3300	230
$GoogleAPI - AuthenticationService$	140	180	150
$GetNotAvailableLayers$	92	110	99
$UpdateUserActivity$	45	150	49
$GetUserSettingsList$	45	450	75

Tabulka 12: Výsledky Posuv času testu z Visual Studia - po úpravách rozhraní

Název dotazů	$min[ms]$	$max[ms]$	$avg[ms]$
$GetStationSPAsJSON$	70	1860	700
$Geoserver$	140	430	200
$GoogleAPI - AuthenticationService$	54	190	82
$GetNotAvailableLayers$	70	110	80
$UpdateUserActivity$	41	200	85
$SetUserSettingsList$	39	44	41
$LoginToSystem$	43	70	56
$LogoutFromSystem$	38	59	41
$GetUserSettingsList$	41	70	52

Tabulka 13: Výsledky Login testu z Visual Studia - po úpravách rozhraní

5.2 Zátěžový test pro 200 uživatelů

Po úpravách bylo vykonáno testování na 200 uživatelích. Byly spuštěny Posuv času, Základní a Login testovací plány.

Název dotazů	minimální čas v ms	průměrný čas v ms	maximální čas v ms
Hydrogram 1 před cache	142	68438	133708
Hydrogram 1 po cache	25	19673	86486
GetSPAsJSON	25	398	6167
$base/OvanetCams$	552	3650	19376
$GetBaseMap$	27	507	10690
$GetDataFromCokies$	3	129	3160
$GetMap$	2	536	15948
$GetMap - flood\_layer$	5	641	24787
$GetMap - Minimap$	7	806	14883
$GetMap - rivers\_network$	5	1318	47548
$GetMap - roads\_network$	5	1208	60415
$GetMap - traffic\_fcd$	5	579	22048
$GetMap - WMS$	3	171	16306
$GetNotAvailableLayers$	1	31	108
$GetUserSettingsList$	2	111	2782
$UpdateUserActivity$	0	15	6646

Tabulka 14: Testování po úpravách z programu JMeter pro 200 uživatelů

Z výsledků uvedených v tabulkách 14, 15 třech testovacích plánů ve dvou testovacích prostředích je zřejmé, že výpočet hydrogramu zabíral celý výkon počítače a že nacachované hydrogramy byly 3.5 x rychlejší než nenacachované hydrogramy. Úprava cachování algoritmu dotazu GetSPAsJSON v testu ukázala, že vrácení dat pro dotaz na GetSPAsJSON trval v průměru 61 milisekund v testovacím prostředí Visual Studio a 398 milisekund v testovacím prostředí JMeter, což přináší dostatečně rychlý čas odezvy.

Název dotazů	minimální čas v ms	průměrný čas v ms	maximální čas v ms
Hydrogram 1	230	11800	78400
GetSPAsJSON	31	61	1670
$OvanetCams$	740	3560	10400
$Geoserver$	140	1010	4170
$GoogleAPI - Auth...$	140	150	240
$GetNotAvailableLayers$	81	130	640
$UpdateUserActivity$	40	57	140
$GetUserSettingsList$	42	660	1660

Tabulka 15: Testování po úpravách z programu Visual Studio pro 200 uživatelů

Výsledky uvedené v tabulkách pro nové rozvržení systému Floreon+ ukázaly, že i při vysokém zatížení systému jsou jednotlivé mapové vrstvy v průměru vráceny do 1000 milisekund pro tile (obrázek). Výsledek rychlosti odpovědi na mapové vrstvy v novém rozvržení systému Floreon+ v testu pro 200 uživatelů odpovídá výsledkům testů pro 50 uživatelů starého rozvržení systému Floreon+. Nejpomalejšími mapovými vrstvami z výsledků testů byly dotazy na dopravní a říční síť, které jsou velmi náročné na paměť Geoserveru, protože obsahují velký objem dat. Je patrné, že při zvýšení zatížení systému čtyřnásobně bylo nové rozvržení systému Floreon+ dostačující.

5.3 Další možné úpravy pro zlepšení výkonu

Jako vhodnou úpravu ke snížení vytížení hardwarových nároků serveru by bylo vhodné dopracovat cachování statických vrstev mapy. V době provádění testů byla cachovaná pouze podkladová mapa. Mapové vrstvy jako dopravní síť, říční síť a další statické vrstvy by měly být cachované pro následné urychlení času odezvy v případě požadavků a hlavně z důvodu snížení nároků na procesor a paměť serveru, kde je Geoserver aplikován.

Dotaz na hydrgoram v testu pro vysoký počet uživatelů v průměru trval 20 sekund pro nacachovaný hydrogram, což představuje dlouhou časovou odezvu. Proto by bylo vhodné otestovat použití jiného cachovacího mechanismu pro hydrogram nebo zpracování dotazu provádět na více serverech.

6 Závěr

Cílem mé bakalářské práce bylo otestovat chování webového rozhraní systému Floreon+ při vysoké uživatelské zátěži a navrhnout možné optimalizační úpravy na snížení přístupové doby. Výsledkem je vytvoření tří testovacích plánů ve dvou testovacích nástrojích v JMeter a Visual Studio, kde měly testovací plány za úkol otestovat větší část webového rozhraní a nalézt úzká místa systému Floreon+. Nikdy nelze otestovat celý systém, ale jen jeho větší část. Testovací balík obsahuje tyto tři testovací plány a je aplikovatelný na jakýkoliv web založený na podobných technologiích.

Vrámci této práce byly navrženy úpravy dle výsledků testovacích plánů. Mezi nalezená úzká místa systému patřil výpočet hydrogramu a stupně povodňové aktivity, neplatné volání části funkcionality a také stahování zdrojových kódů Google API, které byly volány a stahovány z webu. Tyto úpravy byly přínosem a vedly ke zvýšení výkonu systému. Dále byl také otestován vliv výkonu serveru na výpočet hydrogramu. Úprava přenesení výpočtu hydrogramu na server s nižším výkonem se záměrem snížit zátěž už i tak vytíženého serveru, nebyla dobrá volba. Tato modifikace vedla k výraznému zhoršení odezvy i přesto, že tento server byl dedikován jen pro výpočet hydrogramu. Nakonec bylo pro systém vytvořeno nové schéma architektury systému Floreon+, které je připraveno zvládat větší zatížení systému Floreon+.

Největší přínos práce vidím v otestování a nalezení úzkých míst systému. Došlo k jednomu z prvních velkých otestování systému Floreon+. Většina programátorů očekávala, že jedním z úzkých míst budou informace o stupních povodňových aktivit, ale nikdo nečekal, že úzkým místem bude i zobrazení hydrogramu a nezbytné nároky na výpočet hydrogramu. Přínosem jsou také implementované optimalizace, které snižují odezvu při přístupu velkého množství uživatelů.

Existuje několik dalších námětů na možné rozšíření testování. Jako první bych zmínil otestování dalších funkcí systému Floreon+, ke kterým patří např. zadání, výpočet a zobrazení what-if-analýz. Tato nová funkčnost byla dokončena až po otestování systému a zhodnocení výsledků testů, ale v dalších testovacích iteracích by měla být doplněna.

7 Reference

[1] Floreon+ [cit. 28.3.2015]. Dostupné z: http://floreon.vsb.cz

[2] WMS [cit. 28.3.2015]. Dostupné z: http://www.e-cartouche.ch/content_reg/cartouche/webservice/en/html/unit_wms.html

[3] WFS [cit. 28.3.2015]. Dostupné z: http://www.e-cartouche.ch/content_reg/cartouche/webservice/en/html/unit_wfs.html

[4] Geoserver, Geoserver manuál [cit. 28.3.2015]. Dostupné z: http://geoserver.org/

[5] PostgreSQL [cit. 28.3.2015]. Dostupné z: http://www.postgresql.org/about/

[6] PostGIS, PostGIS manuál [cit. 28.3.2015]. Dostupné z: http://postgis.net/

[7] PostGIS, wiki [cit. 28.3.2015]. Dostupné z: http://wiki.openstreetmap.org/wiki/PostGIS/

[8] MS SQL SERVER [cit. 28.3.2015]. Dostupné z: http://www.dotnetportal.cz/clanek/140/Seznameni-a-instalace-Microsoft-SQL-Serveru

[9] WCF, Microsoft MSDN [cit. 28.3.2015] . *Dostupné z: https://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx*

[10] Asp.NET teorie [cit. 28.3.2015]. Dostupné z: http://www.asp.net/get-started

[11] Proxy, whatismyip [cit. 28.3.2015] . *Dostupné z: https://www.whatismyip.com/what-is-a-proxy/*

[12] HTML, w3schools [cit. 28.3.2015]. Dostupné z: http://www.w3schools.com/html/

[13] CSS, w3schools [cit. 28.3.2015]. Dostupné z: http://www.w3schools.com/css/

[14] Openlayers, Openlayers manuál, *[cit. 28.3.2015]. Dostupné z: http://openlayers.org/two/*

[15] Jquery, Jquery manuál [cit. 28.3.2015]. Dostupné z: https://jquery.com/

[16] Ajax, w3schools [cit. 28.3.2015]. Dostupné z: http://www.w3schools.com/ajax/

[17] Roudenský Petr, Havlíčková Anna, *Řízení kvality softwaru Průvodce testováním*, Computer Press Brno, 2013.

[18] Patton Ron, *Testování softwaru*, Computer Press Praha, 2002.

[19] Zátěžové testování [cit. 9.4.2015]. Dostupné z: `http://agiletesting.blogspot.cz/2005/02/performance-vs-load-vs-stress-testing.html/`

[20] JMeter, JMeter manuál [cit. 9.4.2015]. Dostupné z: `http://jmeter.apache.org/`

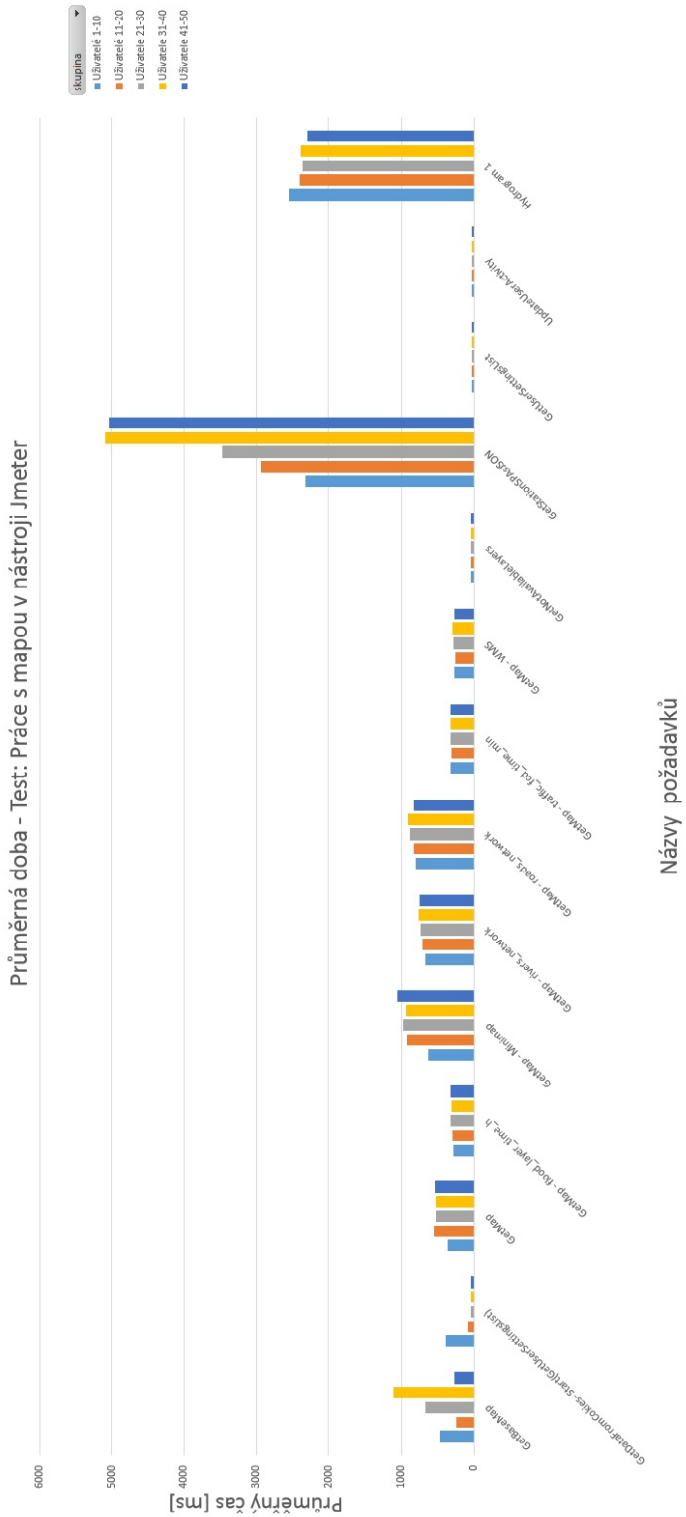
[21] Visual Studio [cit. 9.4.2015]. Dostupné z: `https://www.visualstudio.com/`

Přílohy

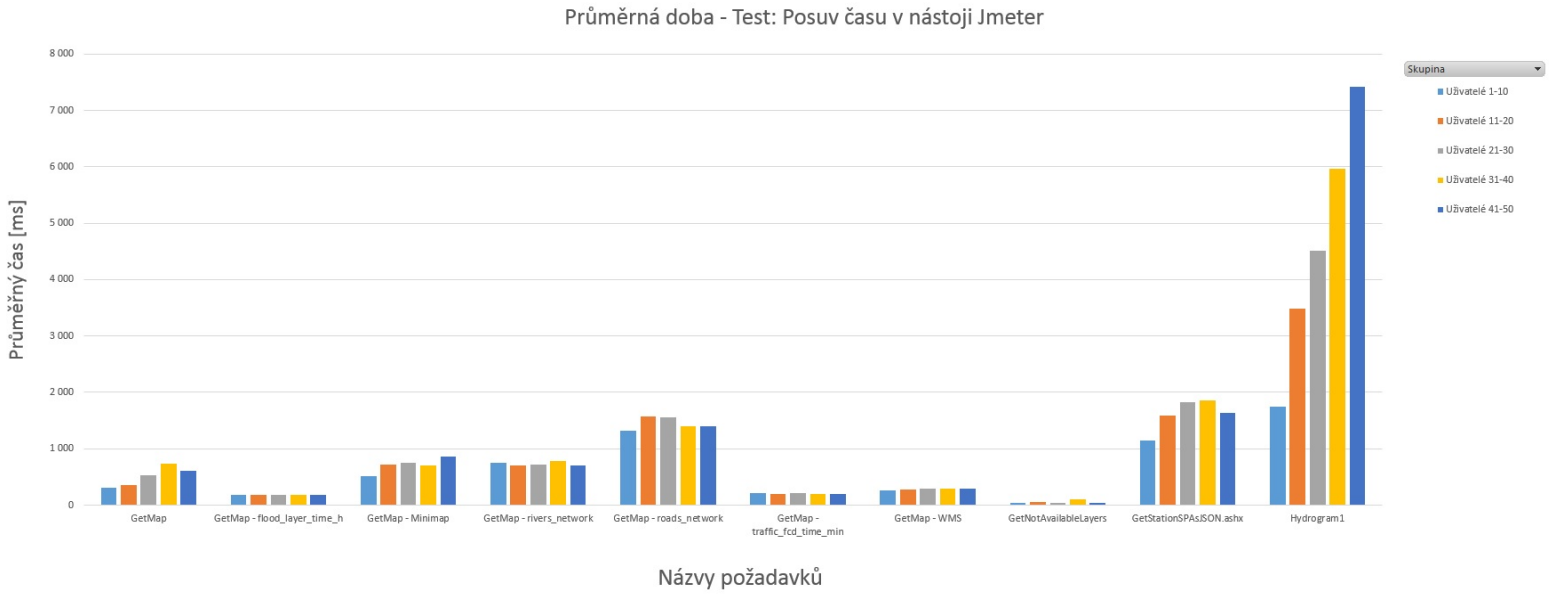
1. Disk DVD obsahuje všechny výsledky testů s grafy v aplikace Microsoft Excel
2. Základní, Posuv času a Login testovací plány pro JMeter – rovněž přiložené na DVD
3. Základní, Posuv času a Login testovací plány pro Visual Studio– rovněž přiložené na DVD

A Grafy

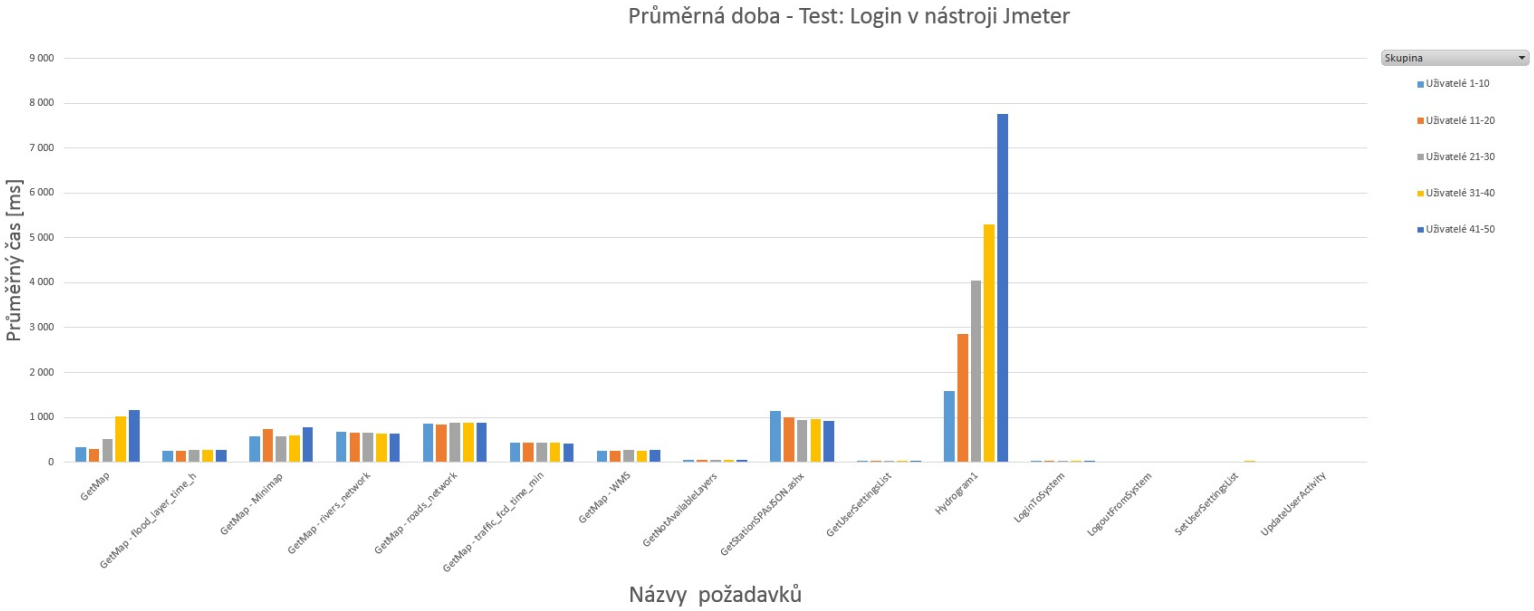
Tato příloha obsahuje grafy z výsledků testů k tabulkám.



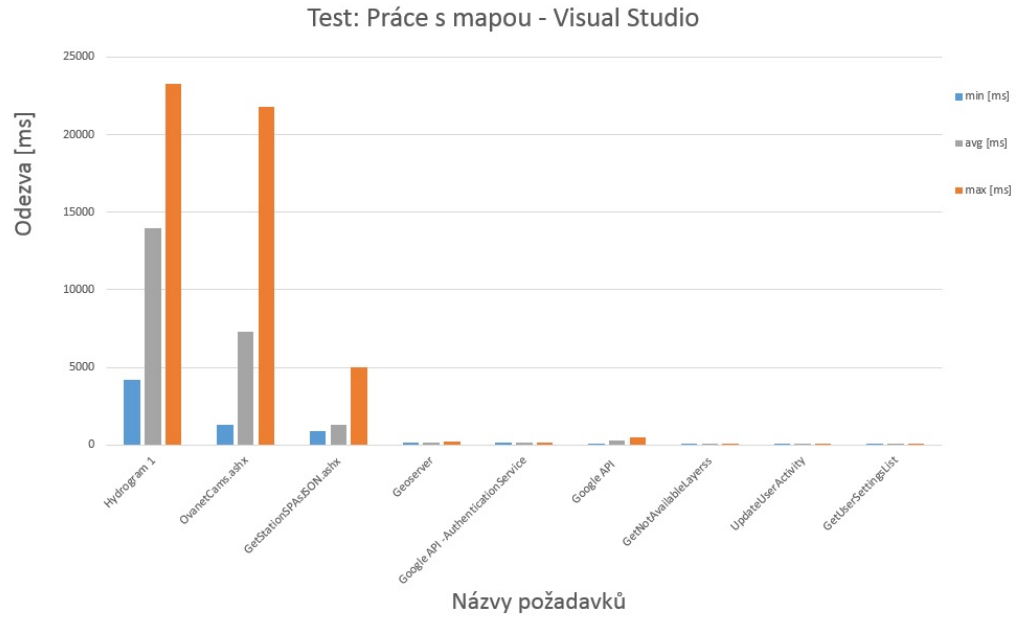
Obrázek 10: Základní test -bez úprav JMeter



Obrázek 11: Posuv času test - bez úprav JMeter

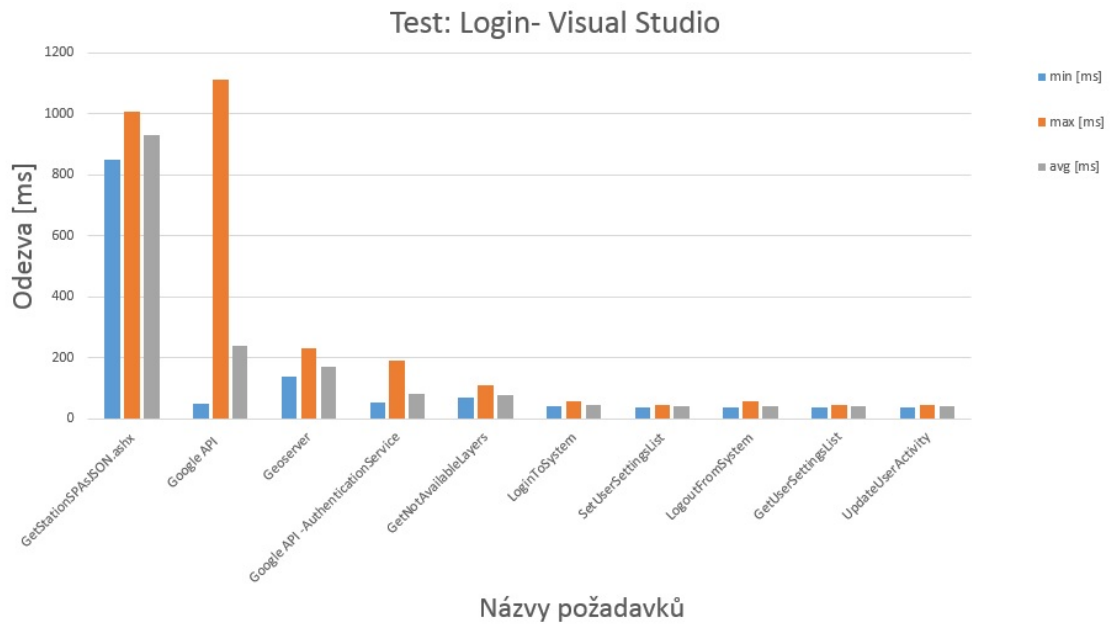


Obrázek 12: Login test - bez úprav JMeter

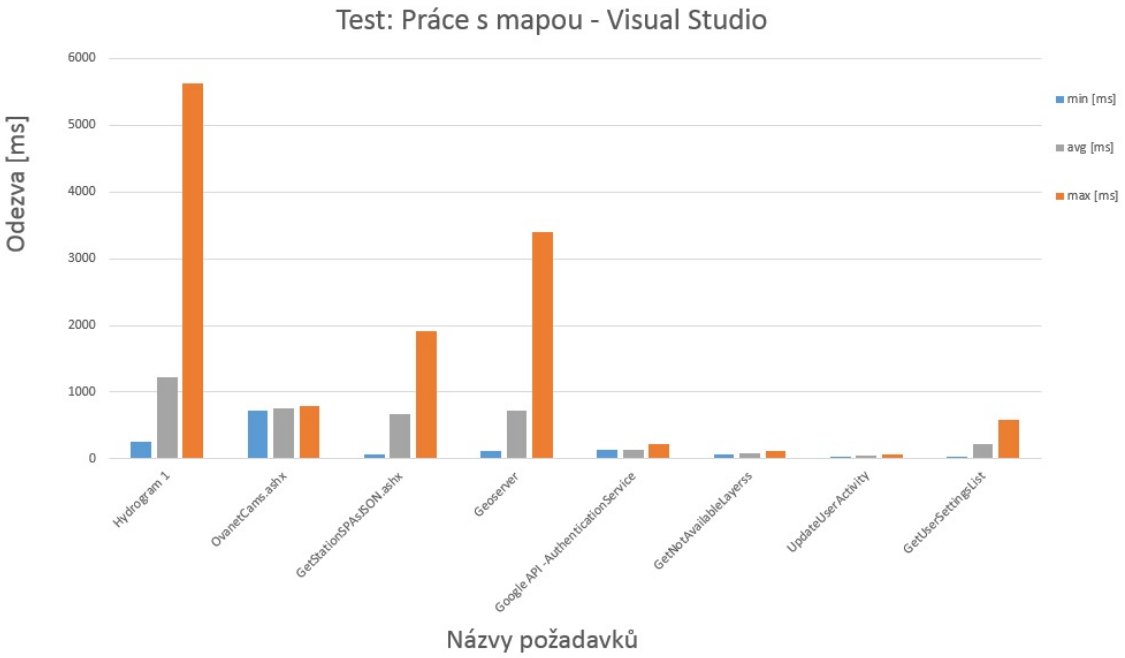


Obrázek 13: Základní test -bez úprav Visual Studio

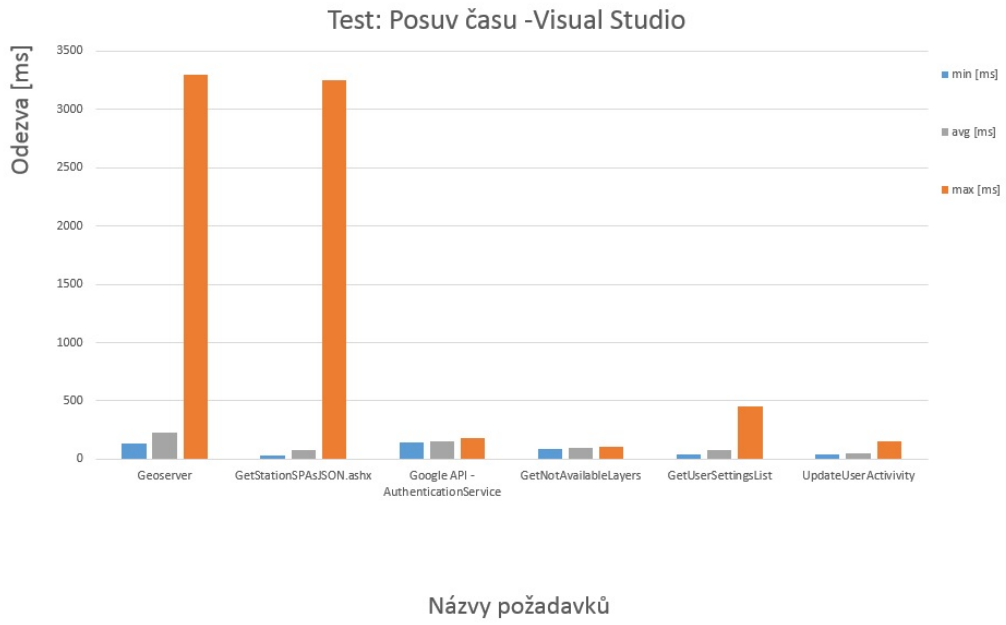
Obrázek 14: Posuv času test - bez úprav Visual Studio



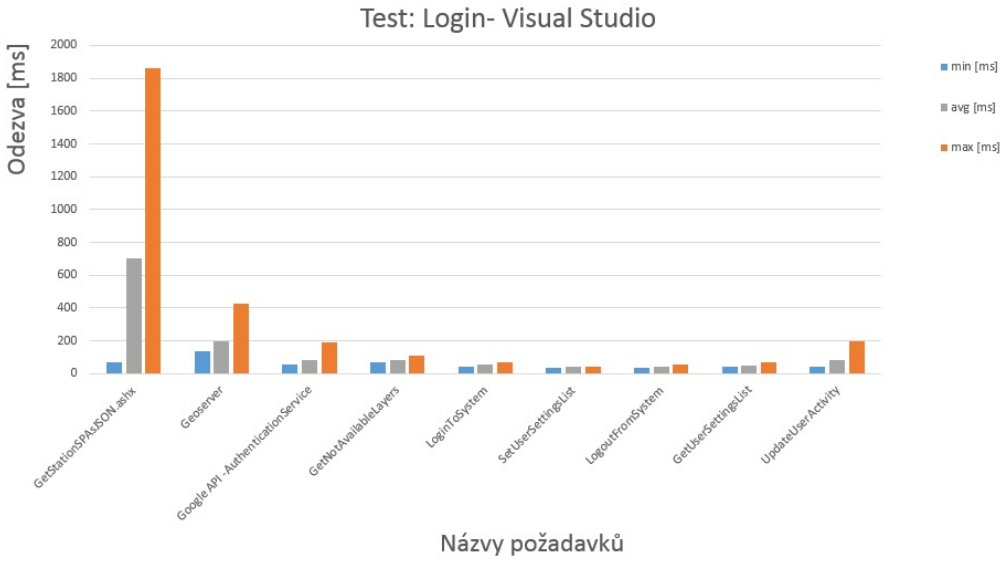
Obrázek 15: Login test - bez úprav Visual Studio



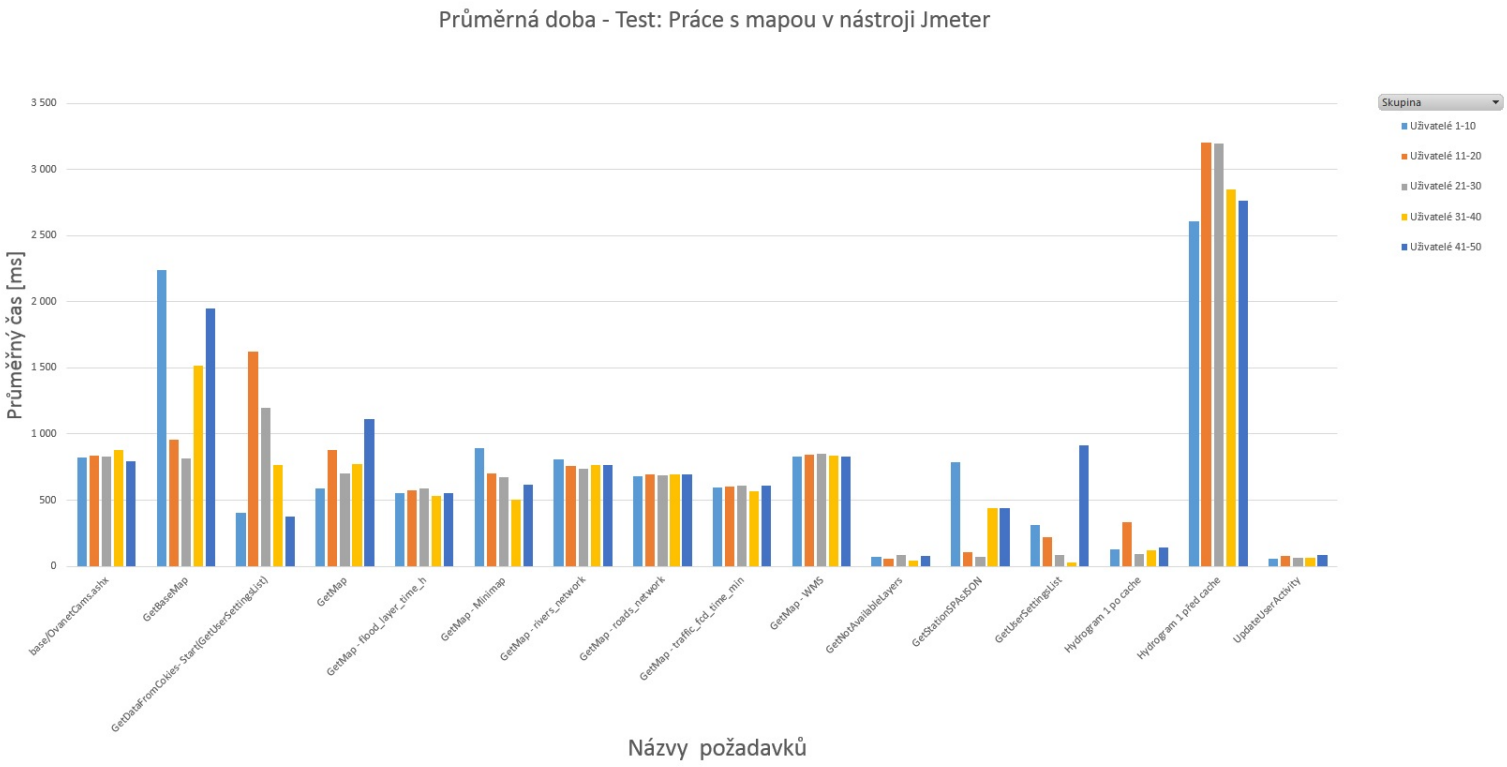
Obrázek 16: Základní test - po úpravách Visual Studio



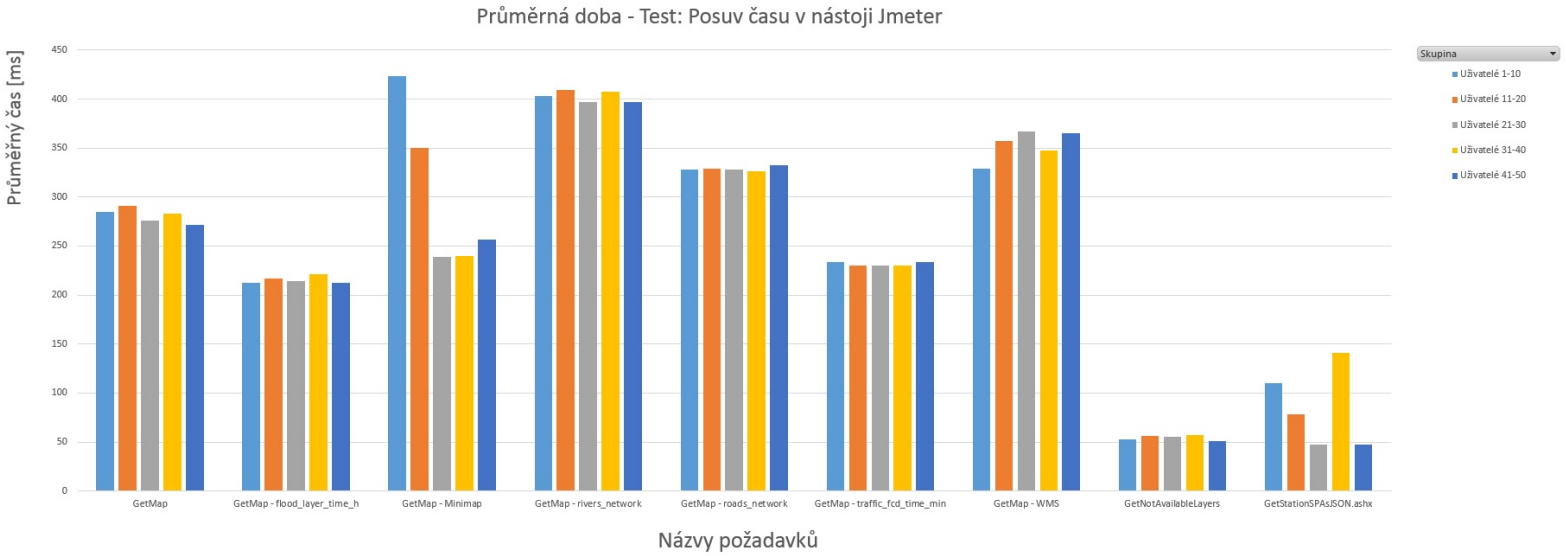
Obrázek 17: Posuv času test - po úpravách Visual Studio



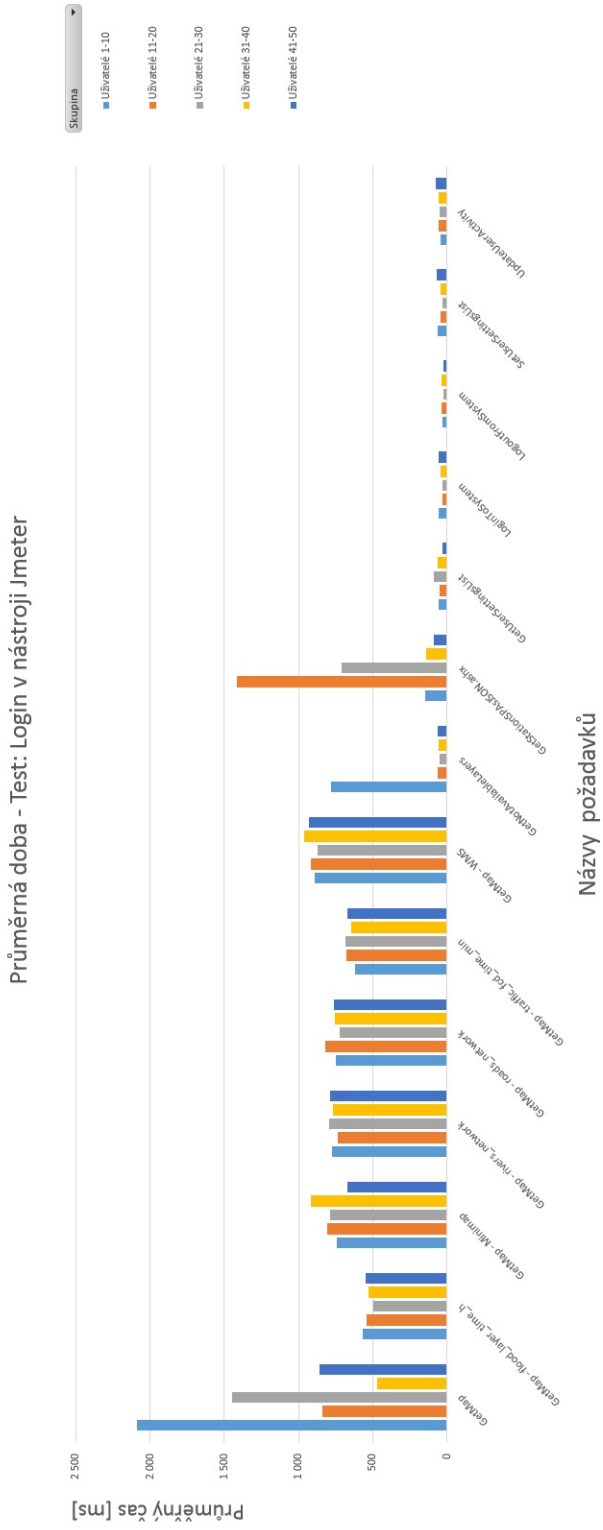
Obrázek 18: Login test - po úpravách Visual Studio



Obrázek 19: Základní test - po úpravách JMeter



Obrázek 20: Posuv času test - po úpravách JMeter



Obrázek 21: Login test - po úpravách JMeter